cholesterol-analysis

November 13, 2024

1 Cholesterol Analysis and Modelling

Dalam proyek ini, kami bertujuan untuk membangun model prediktif faktor-faktor kesehatan apa yang paling berpengaruh terhadap nilai kolesterol total. Kumpulan data berisi sejumlah variabel seperti tekanan darah sistolik dan diastolik, IMT, glukosa puasa, serta trigliserida.

Tahapan Analisis * Data Cleaning * Exploratory Data Analysis * Transformasi data * Model Building * Evaluate Model Performance

1.1 Import Libraries

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import statistics
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
```

1.2 Read Dataset

Dataset ini merupakan data sampel hasil survey kesehatan suatu perusahaan terhadap pegawainya. Dataset ini menyediakan informasi 1339 pegawai dan 15 variabel yang diukur dengan tipe data kategorik dan numerik.

Tujuan analisis ini adalah mencari variabel apa yang paling berpengaruh terhadap tingkat kolesterol seorang pegawai.

Deskripsi Dataset

Index	Variable	Keterangan
1	Jenis kelamin	Variabel yang menunjukkan jenis kelamin responden, laki-laki atau perempuan.
2	Usia	Variabel yang mengindikasikan usia responden dalam tahun.
3	Tekanan darah (S)	Variabel yang mencatat tekanan darah sistolik responden.
4	Tekanan darah (D)	Variabel yang mencatat tekanan darah diastolik responden.
5	Tinggi badan (cm)	Variabel yang mencatat tinggi badan responden dalam sentimeter.
6	Berat badan (kg)	Variabel yang mencatat berat badan responden dalam kilogram.
7	IMT (kg/m2)	Variabel yang menunjukkan Indeks Massa Tubuh (IMT) responden, dihitung sebagai berat badan dalam kilogram dibagi dengan kuadrat tinggi badan dalam meter.
8	Lingkar perut (cm)	Variabel yang mencatat lingkar perut responden dalam sentimeter.
9	Glukosa Puasa (mg/dL)	Variabel yang mencatat kadar glukosa darah puasa responden dalam miligram per desiliter (mg/dL).
10	Cholesterol Total (mg/dL)	Variabel yang mencatat kadar kolesterol total dalam darah responden dalam miligram per desiliter (mg/dL).
11	Trigliserida (mg/dL)	Variabel yang mencatat kadar trigliserida dalam darah responden dalam miligram per desiliter (mg/dL).

Index	Variable	Keterangan
12	Fat	Variabel yang mencatat persentase lemak tubuh responden.
13	Visceral Fat	Persentase lemak visceral (lemak di sekitar organ dalam) dari total lemak tubuh responden.
14	Masa Kerja	Lama masa kerja responden dalam satuan waktu tertentu, misalnya dalam tahun.
15	Tempat lahir	Lokasi tempat lahir responden.

[ ]:
```
path = '/content/drive/MyDrive/ETAM SCIENCE/'
df = pd.read_csv(os.path.join(path, "data.csv"))
df.drop(columns=(['Responden', 'Tempat lahir']), inplace=True)
df.head()
```

[ ]:
```
  Jenis Kelamin  Usia  Tekanan darah (S)  Tekanan darah (D)  \
0             M  19.0              126.0               88.0
1             M  19.0              120.0               80.0
2             M  19.0              120.0               80.0
3             F  19.0              100.0               70.0
4             M  19.0              110.0               70.0

   Tinggi badan (cm)  Berat badan (kg)  IMT (kg/m2)  Lingkar perut (cm)  \
0              172.5              49.5        16.53                66.0
1              158.0              53.6        21.50                71.0
2              170.0              59.5        20.59                80.0
3              149.0              45.1        20.31                62.0
4              171.6              62.4        21.19                78.0

   Glukosa Puasa (mg/dL)  Cholesterol Total (mg/dL)  Trigliserida (mg/dL)  \
0                   84.0                      187.0                  99.0
1                   84.0                      187.0                  99.0
2                   80.0                      187.0                  99.0
3                   81.0                      187.0                  99.0
4                   84.0                      187.0                  99.0

    Fat  Visceral Fat  Masa Kerja
0  26.4           6.0        0.97
1  26.4           6.0        0.60
2  26.4           6.0        1.37
```

```
3  30.5          3.5        1.00
4  26.4          6.0        4.00
```

1.3 Data Cleaning

Melakukan identifikasi dan pembersihan pada dataset yang memiliki missing value, outlier, dan data duplikat.

Identifikasi missing value

Mengidentifikasi missing value pada dataset

[ ]:
```
# Melakukan summary data
column_missing = df.isna().sum().rename('number_of_missing')
column_value = df.apply(lambda x: set(x.unique())).rename('set_of_value')
column_max = df.max(numeric_only=True).rename('maximum')
column_type = df.dtypes.rename('data_type')

summary_df = pd.concat([column_missing, column_value, column_max, column_type],␣
↪axis=1)
summary_df['number_of_value'] = summary_df['set_of_value'].apply(lambda x:␣
↪len(x))
summary_df
```

[ ]:
```
                          number_of_missing  \
Jenis Kelamin                             0
Usia                                      0
Tekanan darah (S)                         0
Tekanan darah (D)                         0
Tinggi badan (cm)                         0
Berat badan (kg)                          0
IMT (kg/m2)                               0
Lingkar perut (cm)                        0
Glukosa Puasa (mg/dL)                     0
Cholesterol Total (mg/dL)                 0
Trigliserida (mg/dL)                      0
Fat                                       0
Visceral Fat                              0
Masa Kerja                                0

                                                                 set_of_value  \
Jenis Kelamin                                                          {F, M}
Usia                      {19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26…
Tekanan darah (S)         {128.0, 129.0, 130.0, 131.0, 132.0, 138.0, 140…
Tekanan darah (D)         {58.0, 60.0, 61.0, 62.0, 63.0, 64.0, 65.0, 66…
Tinggi badan (cm)         {172.4, 170.2, 161.8, 164.3, 164.8, 138.5, 144…
Berat badan (kg)          {58.05, 38.5, 39.85, 40.5, 40.2, 42.4, 43.5, 4…
IMT (kg/m2)               {23.52, 14.85, 15.8, 16.53, 17.96, 18.33, 19.5…
```

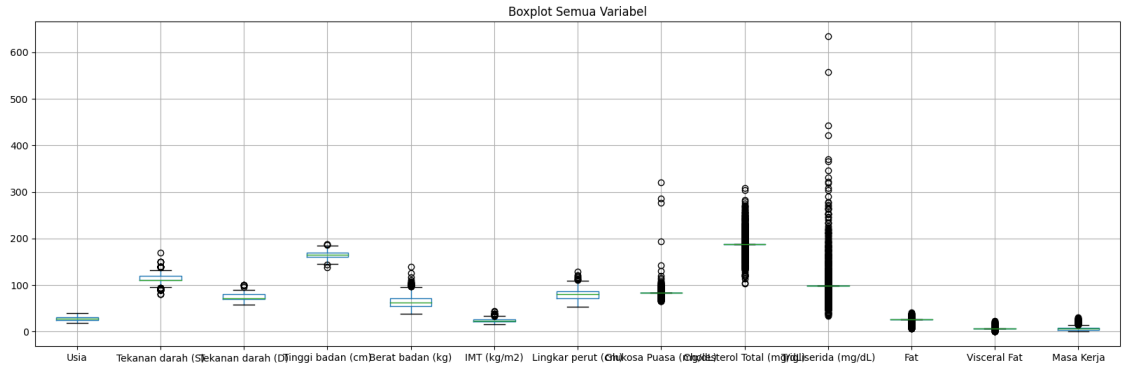
Lingkar perut (cm)	{102.5, 103.0, 104.0, 54.0, 56.0, 57.0, 59.0, …
Glukosa Puasa (mg/dL)	{130.0, 143.0, 321.0, 277.0, 285.0, 68.0, 65.0…
Cholesterol Total (mg/dL)	{103.0, 104.0, 114.0, 118.0, 119.0, 120.0, 122…
Trigliserida (mg/dL)	{34.0, 35.0, 37.0, 38.0, 39.0, 40.0, 41.0, 43…
Fat	{5.8, 6.8, 7.5, 7.4, 9.5, 10.4, 11.1, 11.9, 13…
Visceral Fat	{0.5, 1.0, 2.0, 3.5, 1.5, 4.5, 6.0, 7.0, 2.5, …
Masa Kerja	{0.97, 1.37, 1.0, 1.01, 4.0, 2.0, 6.0, 2.81, 8…

	maximum	data_type	number_of_value
Jenis Kelamin	NaN	object	2
Usia	39.00	float64	21
Tekanan darah (S)	170.00	float64	35
Tekanan darah (D)	100.00	float64	34
Tinggi badan (cm)	187.50	float64	213
Berat badan (kg)	139.75	float64	664
IMT (kg/m2)	44.10	float64	733
Lingkar perut (cm)	128.00	float64	95
Glukosa Puasa (mg/dL)	321.00	float64	50
Cholesterol Total (mg/dL)	308.00	float64	144
Trigliserida (mg/dL)	634.00	float64	187
Fat	40.90	float64	177
Visceral Fat	23.00	float64	43
Masa Kerja	31.00	float64	237

Identifikasi Outliers

dilakukan identifikasi outlier pada setiap variabel dataset dan menghitung jumlah outlier yang terbentuk.

[ ]:
```
df.boxplot(figsize=(20, 6))
plt.title('Boxplot Semua Variabel')
plt.show()
```



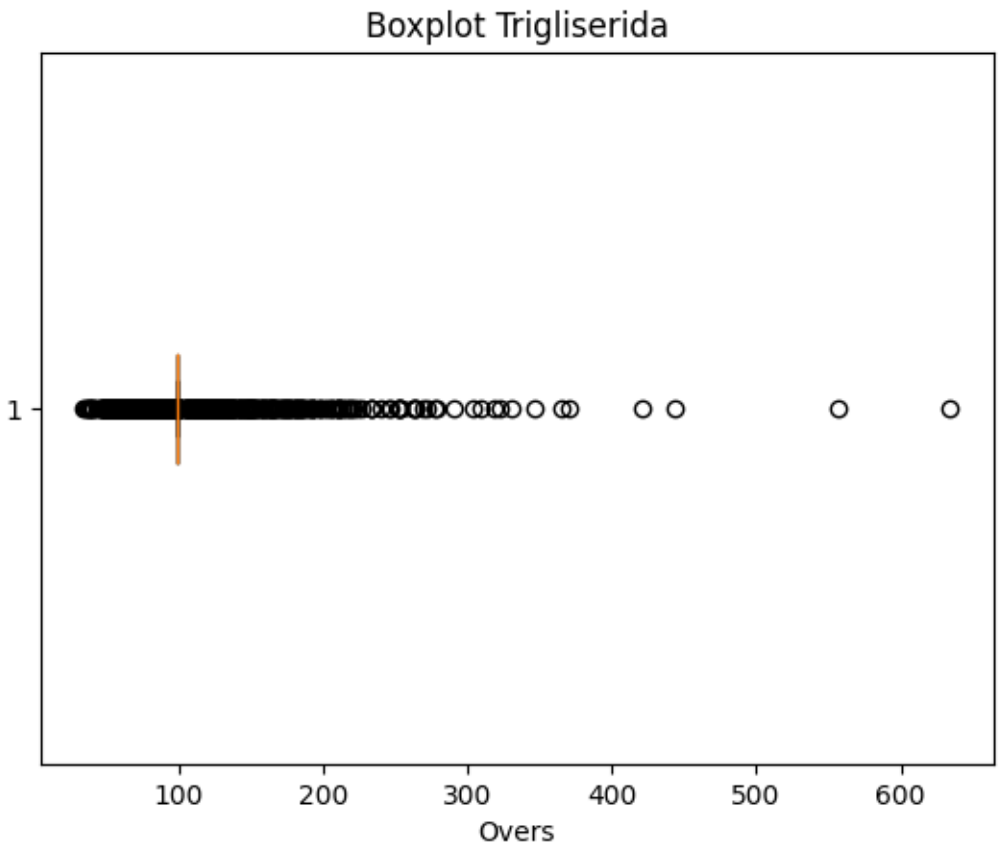
[ ]:
```
# Fungsi Pendeteksian Outlier dengan IQR
def detect_outliers_iqr(data):
    outliers = []
    data = sorted(data)
    q1 = np.percentile(data, 25)
    q3 = np.percentile(data, 75)

    IQR = q3-q1
    lwr_bound = q1-(1.5*IQR)
    upr_bound = q3+(1.5*IQR)

    for i in data:
        if (i<lwr_bound or i>upr_bound):
            outliers.append(i)
    return outliers
sample_outliers = detect_outliers_iqr(df['Trigliserida (mg/dL)'])
# print("Outliers from IQR method: ", sample_outliers)
len(sample_outliers)
# sample_outliers
```

[ ]:
```
548
```

Mengidentifikasi jumlah outlier pada salah satu variabel yaitu variabel trigliserida

[ ]:
```
plt.boxplot(df['Trigliserida (mg/dL)'], vert=False)
plt.title("Boxplot Trigliserida")
plt.xlabel('Overs')
```

[ ]:
```
Text(0.5, 0, 'Overs')
```



Identifikasi data duplikat

Mengidentifikasi data yang duplikat pada dataset

[ ]:
```
df.duplicated(keep=False).sum()
```

[ ]:
```
0
```

1.4 Exploratory Data Analysis

Statistika Deskriptif

Menampilkan tabel statistika deskriptif pada seleuruh variabel dataset

[ ]:
```
#statistika deskriptif
df.describe()
```

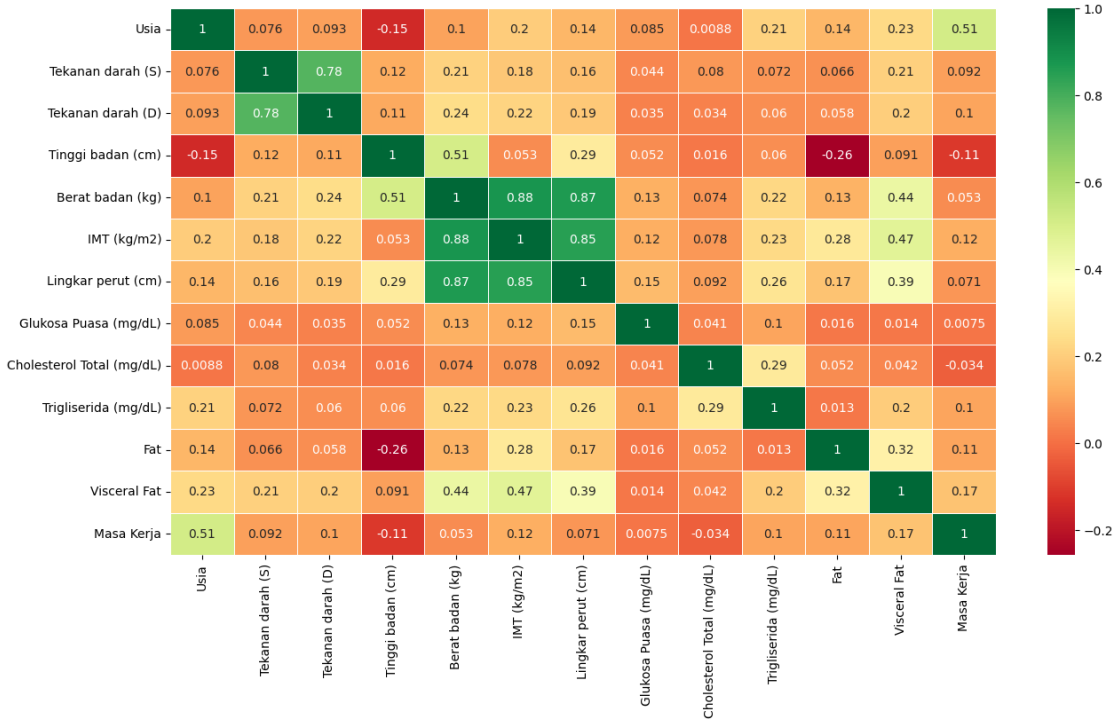
[ ]:
```
              Usia  Tekanan darah (S)  Tekanan darah (D)  Tinggi badan (cm)  \
count  1339.000000        1339.000000        1339.000000        1339.000000
mean     28.597461         113.147872          74.009709         164.940851
std       4.767230          10.164592           7.718752           7.386617
```

```
min      19.000000          80.000000          58.000000         138.500000
25%      25.000000         110.000000          70.000000         160.000000
50%      28.000000         110.000000          72.000000         165.000000
75%      31.000000         120.000000          80.000000         170.000000
max      39.000000         170.000000         100.000000         187.500000

       Berat badan (kg)  IMT (kg/m2)  Lingkar perut (cm)  \
count       1339.000000  1339.000000         1339.000000
mean          64.620500    23.693727           80.441972
std           12.799096     4.021585           10.688215
min           38.500000    14.850000           54.000000
25%           55.275000    20.855000           72.000000
50%           62.500000    23.200000           80.000000
75%           71.775000    26.000000           87.000000
max          139.750000    44.100000          128.000000

       Glukosa Puasa (mg/dL)  Cholesterol Total (mg/dL)  Trigliserida (mg/dL)  \
count            1339.000000                1339.000000           1339.000000
mean               84.571322                 187.995519            106.982823
std                11.522057                  21.104834             44.143456
min                65.000000                 103.000000             34.000000
25%                84.000000                 187.000000             99.000000
50%                84.000000                 187.000000             99.000000
75%                84.000000                 187.000000             99.000000
max               321.000000                 308.000000            634.000000

                Fat  Visceral Fat   Masa Kerja
count  1339.000000   1339.000000  1339.000000
mean     26.203510      6.231367     6.401837
std       3.678467      2.431923     4.554438
min       5.800000      0.500000     0.000000
25%      26.400000      6.000000     4.000000
50%      26.400000      6.000000     6.000000
75%      26.400000      6.000000     8.000000
max      40.900000     23.000000    31.000000
```

Matriks Korelasi

Menampilkan matriks korelasi pada seluruh variabel

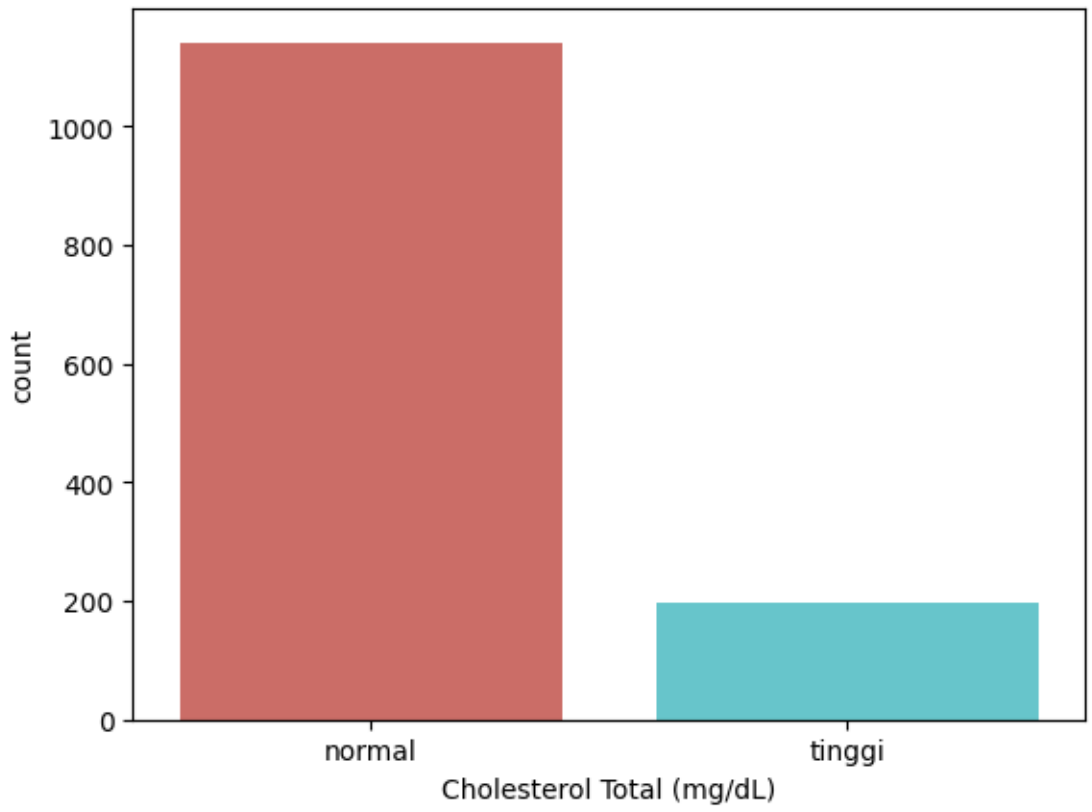
[ ]:
```
df_numeric = df.select_dtypes(include=[np.number])
corr_matrix = df_numeric.corr()

sns.heatmap(corr_matrix, linewidths=0.5, annot=True, cmap='RdYlGn')
plt.gcf().set_size_inches(15, 8)
plt.show()
```



[ ]:
```
# Variabel Y
df['Cholesterol Total (mg/dL)'] = df['Cholesterol Total (mg/dL)'].apply(lambda␣
 ↪x : 'normal' if x < 200  else 'tinggi')

# Trigliserida
df['Trigliserida (mg/dL)'] = df['Trigliserida (mg/dL)'].apply(lambda x:␣
 ↪'normal' if x < 150 else 'tinggi')
```

[ ]:
```
sns.countplot(x='Cholesterol Total (mg/dL)',data=df, palette='hls')
plt.show()
```

```
<ipython-input-12-f5a811f2ef8d>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Cholesterol Total (mg/dL)',data=df, palette='hls')
```



[ ]:
```
df['Cholesterol Total (mg/dL)'].value_counts()
```

[ ]:
```
Cholesterol Total (mg/dL)
normal    1141
tinggi     198
Name: count, dtype: int64
```

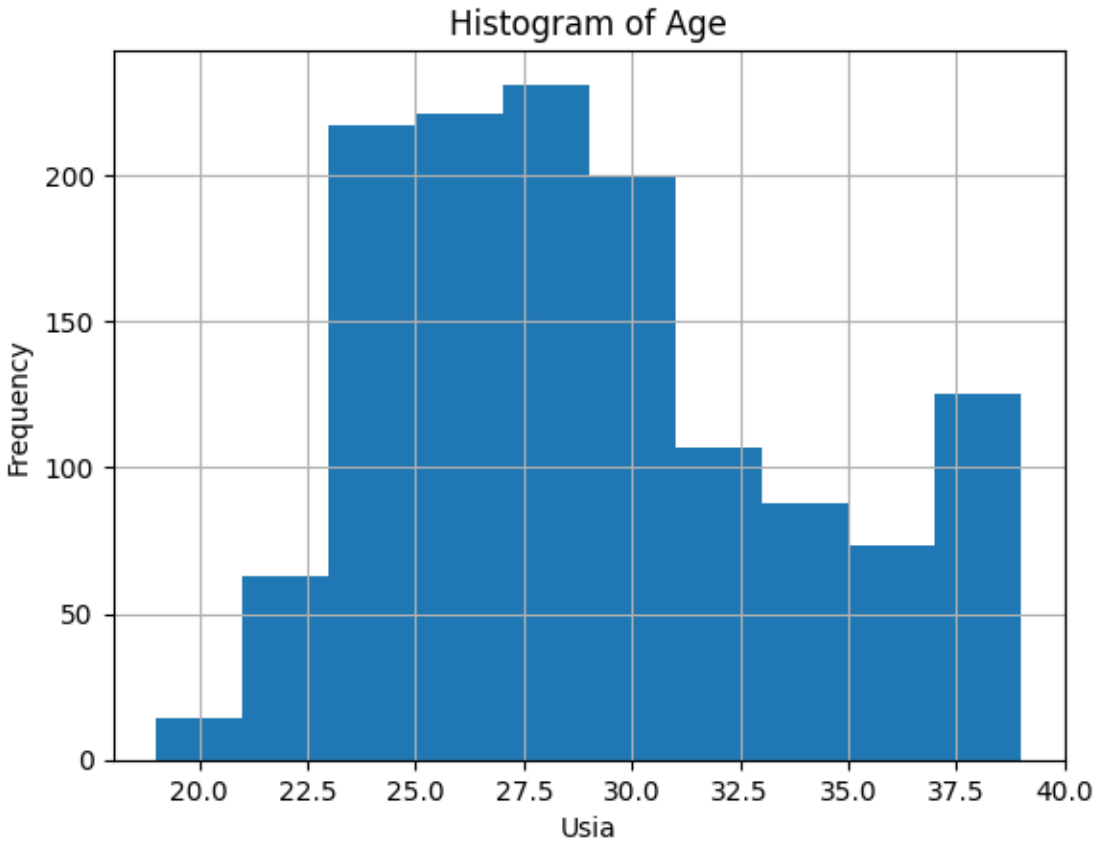
[ ]:
```
selected_columns = ['Usia', 'Tinggi badan (cm)', 'Berat badan (kg)', 'IMT (kg/
↪m2)', 'Glukosa Puasa (mg/dL)',
                    'Lingkar perut (cm)', 'Masa Kerja']
grouped_mean = df.groupby('Cholesterol Total (mg/dL)')[selected_columns].mean()
print(grouped_mean)
```

```
                              Usia  Tinggi badan (cm)  Berat badan (kg)  \
Cholesterol Total (mg/dL)
normal                   27.987730         164.972656         64.203129
tinggi                   32.111111         164.757576         67.025657

                         IMT (kg/m2)  Glukosa Puasa (mg/dL)  \
Cholesterol Total (mg/dL)
normal                     23.536398              84.377739
```

```
tinggi                          24.600354              85.686869

                                Lingkar perut (cm)  Masa Kerja
Cholesterol Total (mg/dL)
normal                                  80.059159    6.179395
tinggi                                  82.647980    7.683687
```

[ ]:
```
df.Usia.hist()
plt.title('Histogram of Age')
plt.xlabel('Usia')
plt.ylabel('Frequency')
```

[ ]:
```
Text(0, 0.5, 'Frequency')
```

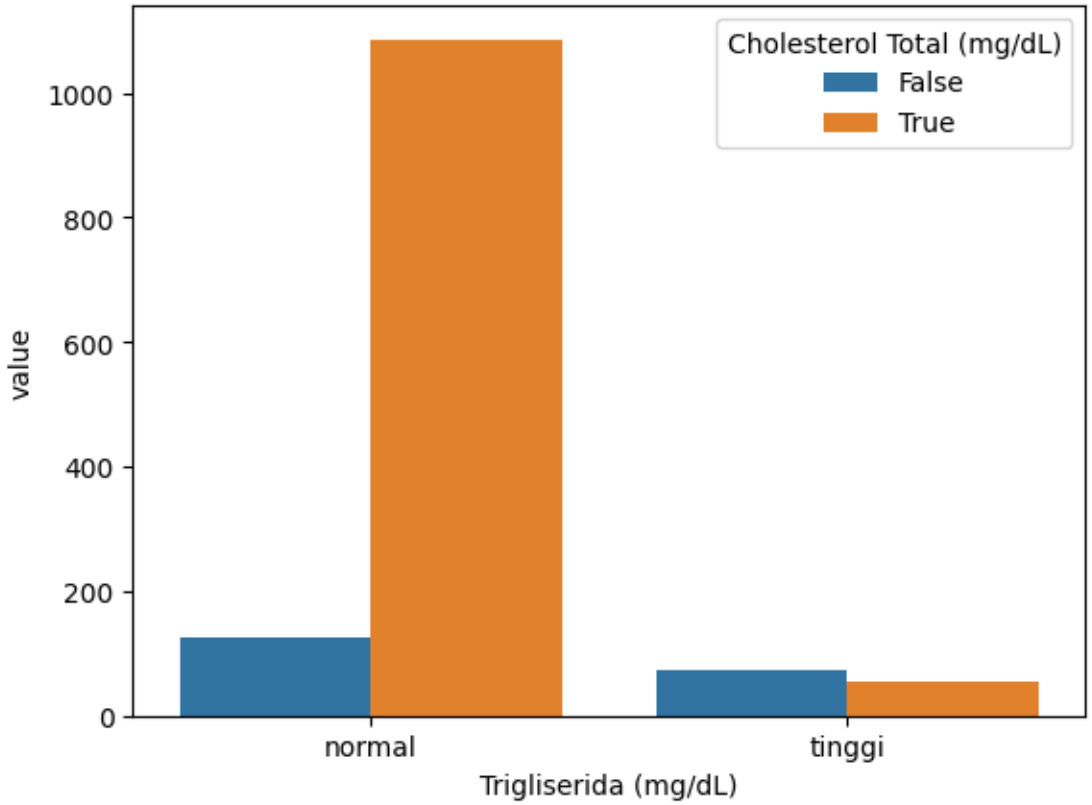


Membandingkan variabel cholesterol dengan trigliserida

Untuk mengetahui apakah jika trigliserida masuk dalam kategori normal maka juga memiliki tingkat kolesterol normal atau tidak

[ ]:
```
col_y = pd.crosstab(df['Trigliserida (mg/dL)'], (df['Cholesterol Total (mg/
↪dL)'] == "normal"))
```

```
loan_stacked = col_y.stack().reset_index().rename(columns={0:'value'})
sns.barplot(x='Trigliserida (mg/dL)', y='value', hue='Cholesterol Total (mg/
 ↪dL)', data=loan_stacked)
plt.show()
```



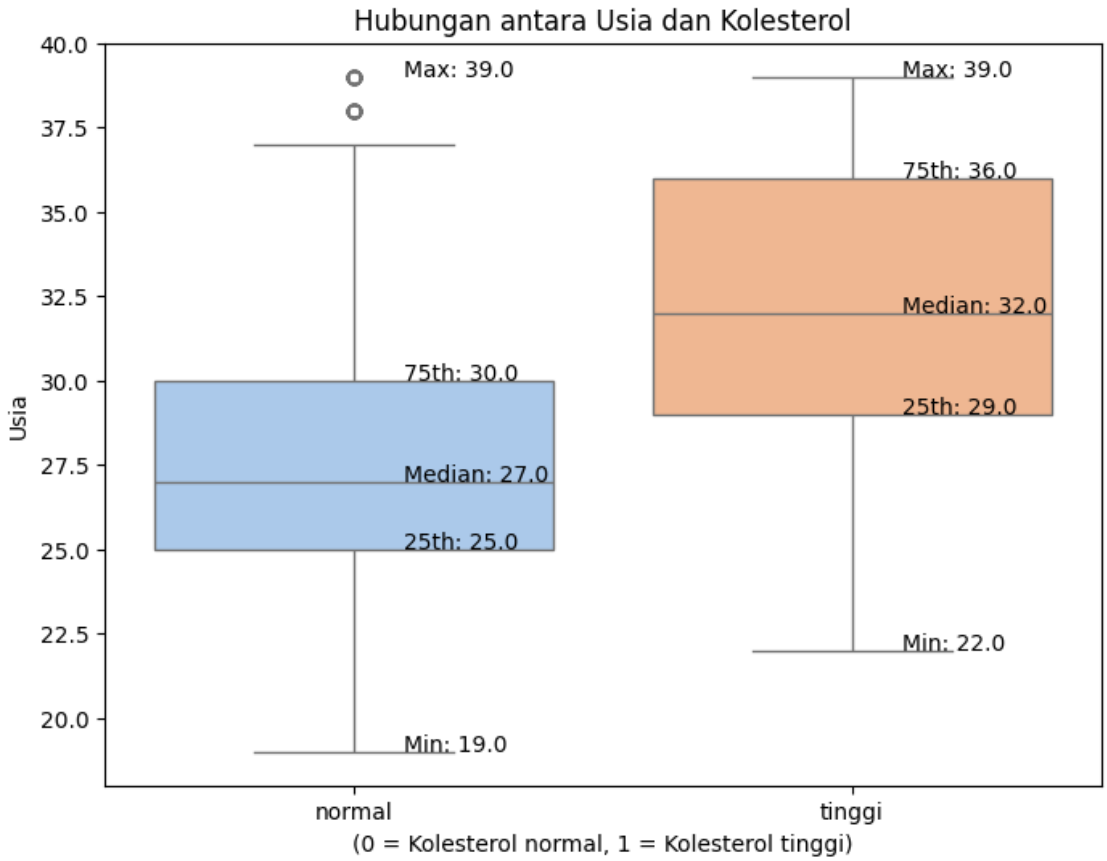
Boxplot

Menampilkan boxplot variabel usia dan kolesterol untuk mengetahui rentang usia pada variabel kolesterol

[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot Usia berdasarkan kategori kolestrol
plt.figure(figsize=(8, 6))
sns.boxplot(x='Cholesterol Total (mg/dL)', y='Usia', data=df, palette='pastel')

# Menambahkan nilai pada boxplot
for i in range(len(df['Cholesterol Total (mg/dL)'].unique())):
    box_data = df[df['Cholesterol Total (mg/dL)'] == df['Cholesterol Total (mg/
 ↪dL)'].unique()[i]]['Usia']
```

```
    whisker_bottom = box_data.min()
    whisker_top = box_data.max()
    median = box_data.median()
    quartile_25 = box_data.quantile(0.25)
    quartile_75 = box_data.quantile(0.75)

    plt.annotate(f'Min: {whisker_bottom}', xy=(i, whisker_bottom), xytext=(i+0.
↪1, whisker_bottom), color='black')
    plt.annotate(f'25th: {quartile_25}', xy=(i, quartile_25), xytext=(i+0.1,␣
↪quartile_25), color='black')
    plt.annotate(f'Median: {median}', xy=(i, median), xytext=(i+0.1, median),␣
↪color='black')
    plt.annotate(f'75th: {quartile_75}', xy=(i, quartile_75), xytext=(i+0.1,␣
↪quartile_75), color='black')
    plt.annotate(f'Max: {whisker_top}', xy=(i, whisker_top), xytext=(i+0.1,␣
↪whisker_top), color='black')

plt.title('Hubungan antara Usia dan Kolesterol')
plt.xlabel('(0 = Kolesterol normal, 1 = Kolesterol tinggi)')
plt.ylabel('Usia')
plt.show()
```

```
<ipython-input-17-ae7526c66e7c>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='Cholesterol Total (mg/dL)', y='Usia', data=df,
palette='pastel')
```



1.5 Transformasi data

Dilakukan penanganan pada data kategorik dengan membuat data dummy sesuai dengan kategori yang telah ditentukan

Dilakukan labeling data sesuai dengan kategori berikut

Index	Variable	Keterangan
1	Jenis kelamin	0 = wanita, 1 = pria
2	Usia	00 = <=35 tahun, 10 = 36-50 tahun, 01 = >50 tahun
3	IMT (kg/m2)	000 = kurus(<18.5), 100 = normal(18.5-24), 010 = kegemukan(25-26), 001 = obesitas(>26)
4	Glukosa Puasa (mg/dL)	0 = normal (100), 1 = pra-diabetes/tidak normal (>100)
5	Cholesterol Total (mg/dL)	0 = normal (<200), 1 = tinggi (200)
6	Trigliserida (mg/dL)	0 = normal (<150), 1 = tinggi (150)

[ ]:
```
# Jenis Kelamin
df['Jenis Kelamin'] = df['Jenis Kelamin'].apply(lambda x: 'Wanita' if x == 'F'␣
 ↪else 'Pria')

# Usia
df['Usia'] = df['Usia'].apply(lambda x: '0' if 19 <= x <= 35 else ('1' if 36 <=␣
 ↪x <= 50 else '2'))

# IMT
df['IMT (kg/m2)'] = df['IMT (kg/m2)'].apply(lambda x: '0' if x < 18.5 else ('1'␣
 ↪if 18.5 <= x < 25.0 else ('2' if 25.0 <= x < 27.0 else '3')))

# Glukosa Puasa
df['Glukosa Puasa (mg/dL)'] = df['Glukosa Puasa (mg/dL)'].apply(lambda x: '0'␣
 ↪if x < 100 else '1')
df[['Jenis Kelamin','Cholesterol Total (mg/dL)', 'Glukosa Puasa (mg/
 ↪dL)','Trigliserida (mg/dL)', 'IMT (kg/m2)','Usia']] = df[['Jenis␣
 ↪Kelamin','Cholesterol Total (mg/dL)', 'Glukosa Puasa (mg/dL)','Trigliserida␣
 ↪(mg/dL)', 'IMT (kg/m2)','Usia']] .astype('category')
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1339 entries, 0 to 1338
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Jenis Kelamin              1339 non-null   category
 1   Usia                       1339 non-null   category
 2   Tekanan darah (S)          1339 non-null   float64
 3   Tekanan darah (D)          1339 non-null   float64
 4   Tinggi badan (cm)          1339 non-null   float64
 5   Berat badan (kg)           1339 non-null   float64
 6   IMT (kg/m2)                1339 non-null   category
 7   Lingkar perut (cm)         1339 non-null   float64
 8   Glukosa Puasa (mg/dL)      1339 non-null   category
 9   Cholesterol Total (mg/dL)  1339 non-null   category
 10  Trigliserida (mg/dL)       1339 non-null   category
 11  Fat                        1339 non-null   float64
 12  Visceral Fat               1339 non-null   float64
 13  Masa Kerja                 1339 non-null   float64
dtypes: category(6), float64(8)
memory usage: 92.5 KB
```

Membuat Data Dummy

Data dummy dibuat pada variabel kategori yaitu usia, IMT, jenis kelamin, glukosa puasa, cholesterol total, dan trigliserida

[ ]:
```
cat_vars = ['Usia', 'IMT (kg/m2)', 'Jenis Kelamin','Glukosa Puasa (mg/dL)',␣
 ↪'Cholesterol Total (mg/dL)','Trigliserida (mg/dL)']
for var in cat_vars:
    cat_list = pd.get_dummies(df[var], prefix=var)
    df = pd.concat([df, cat_list], axis=1)
df.drop(cat_vars, axis=1, inplace=True)
print(df.columns.values)
```

```
['Tekanan darah (S)' 'Tekanan darah (D)' 'Tinggi badan (cm)'
 'Berat badan (kg)' 'Lingkar perut (cm)' 'Fat' 'Visceral Fat' 'Masa Kerja'
 'Usia_0' 'Usia_1' 'IMT (kg/m2)_0' 'IMT (kg/m2)_1' 'IMT (kg/m2)_2'
 'IMT (kg/m2)_3' 'Jenis Kelamin_Pria' 'Jenis Kelamin_Wanita'
 'Glukosa Puasa (mg/dL)_0' 'Glukosa Puasa (mg/dL)_1'
 'Cholesterol Total (mg/dL)_normal' 'Cholesterol Total (mg/dL)_tinggi'
 'Trigliserida (mg/dL)_normal' 'Trigliserida (mg/dL)_tinggi']
```

[ ]:
```
df.head()
```

[ ]:

	Tekanan darah (S)	Tekanan darah (D)	Tinggi badan (cm)	Berat badan (kg)	\
0	126.0	88.0	172.5	49.5	
1	120.0	80.0	158.0	53.6	
2	120.0	80.0	170.0	59.5	
3	100.0	70.0	149.0	45.1	
4	110.0	70.0	171.6	62.4	

	Lingkar perut (cm)	Fat	Visceral Fat	Masa Kerja	Usia_0	Usia_1	…	\
0	66.0	26.4	6.0	0.97	True	False	…	
1	71.0	26.4	6.0	0.60	True	False	…	
2	80.0	26.4	6.0	1.37	True	False	…	
3	62.0	30.5	3.5	1.00	True	False	…	
4	78.0	26.4	6.0	4.00	True	False	…	

	IMT (kg/m2)_2	IMT (kg/m2)_3	Jenis Kelamin_Pria	Jenis Kelamin_Wanita	\
0	False	False	True	False	
1	False	False	True	False	
2	False	False	True	False	
3	False	False	False	True	
4	False	False	True	False	

	Glukosa Puasa (mg/dL)_0	Glukosa Puasa (mg/dL)_1	\
0	True	False	
1	True	False	
2	True	False	
3	True	False	
4	True	False	

	Cholesterol Total (mg/dL)_normal	Cholesterol Total (mg/dL)_tinggi	\

```
0                                 True                                 False
1                                 True                                 False
2                                 True                                 False
3                                 True                                 False
4                                 True                                 False

   Trigliserida (mg/dL)_normal  Trigliserida (mg/dL)_tinggi
0                         True                        False
1                         True                        False
2                         True                        False
3                         True                        False
4                         True                        False

[5 rows x 22 columns]
```

1.6 Model Building

Mengubah tipe data object menjadi tipe data integer

data dummy yang telah dibuat memiliki tipe data object, agar bisa dilakukan pembuatan model regresi logistik data harus memiliki tipe data numerik sehingga dilakukan perubahan tipe data object menjadi tipe data integer

[ ]:
```
df = pd.DataFrame(df)
```

[ ]:
```
df[['Tekanan darah (S)','Tekanan darah (D)','Tinggi badan (cm)',
 'Berat badan (kg)','Lingkar perut (cm)','Fat','Visceral Fat','Masa Kerja',
 'Usia_0','Usia_1','IMT (kg/m2)_0','IMT (kg/m2)_1','IMT (kg/m2)_2',
 'IMT (kg/m2)_3','Jenis Kelamin_Pria','Jenis Kelamin_Wanita',
 'Glukosa Puasa (mg/dL)_0','Glukosa Puasa (mg/dL)_1',
 'Cholesterol Total (mg/dL)_normal','Cholesterol Total (mg/dL)_tinggi',
 'Trigliserida (mg/dL)_normal','Trigliserida (mg/dL)_tinggi']] = df[['Tekanan␣
 ↪darah (S)','Tekanan darah (D)','Tinggi badan (cm)',
 'Berat badan (kg)','Lingkar perut (cm)','Fat','Visceral Fat','Masa Kerja',
 'Usia_0','Usia_1','IMT (kg/m2)_0','IMT (kg/m2)_1','IMT (kg/m2)_2',
 'IMT (kg/m2)_3','Jenis Kelamin_Pria','Jenis Kelamin_Wanita',
 'Glukosa Puasa (mg/dL)_0','Glukosa Puasa (mg/dL)_1',
 'Cholesterol Total (mg/dL)_normal','Cholesterol Total (mg/dL)_tinggi',
 'Trigliserida (mg/dL)_normal','Trigliserida (mg/dL)_tinggi']].astype(int)
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1339 entries, 0 to 1338
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Tekanan darah (S)                1339 non-null   int64
```

```
 1   Tekanan darah (D)                 1339 non-null   int64
 2   Tinggi badan (cm)                 1339 non-null   int64
 3   Berat badan (kg)                  1339 non-null   int64
 4   Lingkar perut (cm)                1339 non-null   int64
 5   Fat                               1339 non-null   int64
 6   Visceral Fat                      1339 non-null   int64
 7   Masa Kerja                        1339 non-null   int64
 8   Usia_0                            1339 non-null   int64
 9   Usia_1                            1339 non-null   int64
 10  IMT (kg/m2)_0                     1339 non-null   int64
 11  IMT (kg/m2)_1                     1339 non-null   int64
 12  IMT (kg/m2)_2                     1339 non-null   int64
 13  IMT (kg/m2)_3                     1339 non-null   int64
 14  Jenis Kelamin_Pria                1339 non-null   int64
 15  Jenis Kelamin_Wanita              1339 non-null   int64
 16  Glukosa Puasa (mg/dL)_0           1339 non-null   int64
 17  Glukosa Puasa (mg/dL)_1           1339 non-null   int64
 18  Cholesterol Total (mg/dL)_normal  1339 non-null   int64
 19  Cholesterol Total (mg/dL)_tinggi  1339 non-null   int64
 20  Trigliserida (mg/dL)_normal       1339 non-null   int64
 21  Trigliserida (mg/dL)_tinggi       1339 non-null   int64
dtypes: int64(22)
memory usage: 230.3 KB
```

Split data

Dilakukan split data training dan data testing dengan rasio 70%:30%

[ ]:
```
y = df['Cholesterol Total (mg/dL)_tinggi']
X = df.drop(['Cholesterol Total (mg/dL)_tinggi','Cholesterol Total (mg/
 ↪dL)_normal','Jenis Kelamin_Wanita','Glukosa Puasa (mg/dL)_1',
             'Trigliserida (mg/dL)_tinggi','Usia_1'], axis=1)
```

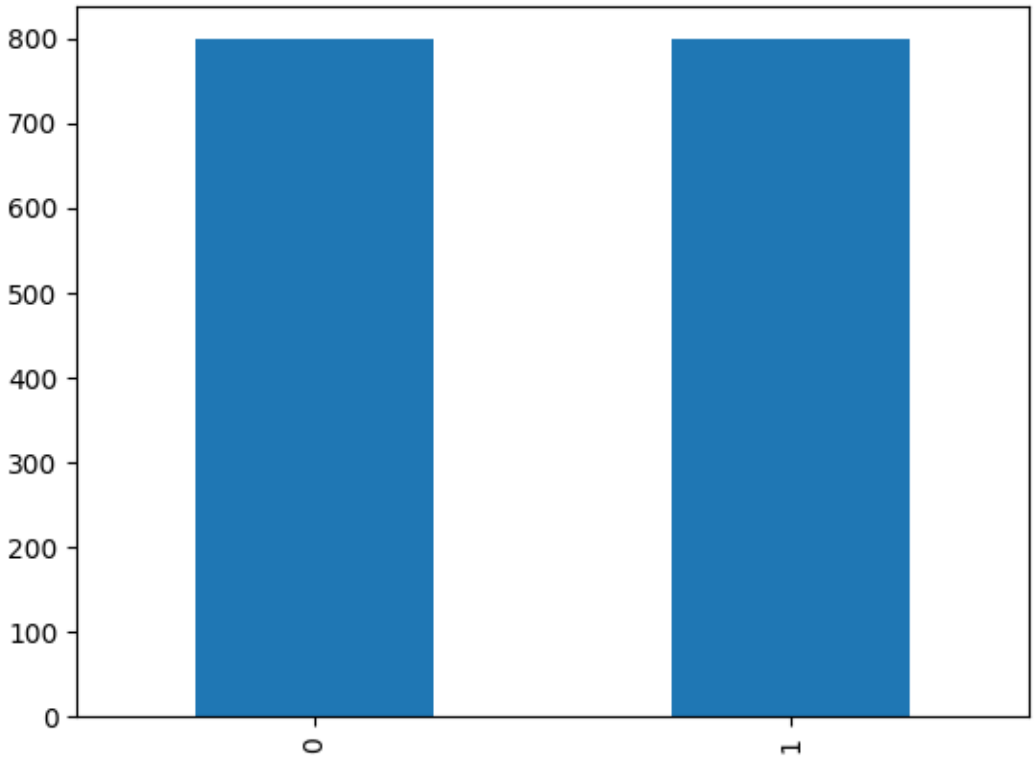
[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
 ↪random_state = 2, shuffle = True, stratify = y)
```

Penangan over-sampling dengan SMOTE

[ ]:
```
# Proses SMOTE
sm = SMOTE(random_state=22)
X_train_new, y_train_new = sm.fit_resample(X_train, y_train.ravel())
```

[ ]:
```
pd.Series(y_train_new).value_counts().plot.bar()
```

[ ]:
```
<Axes: >
```



Implementasi model regresi logistik

dilakukan implementasi regresi logistik pada seluruh variabel, kemudian dilakukan pengecekkan nilai p-value untuk mengetahui variabel mana yang signifikan terhadap model

Tahap 1

[ ]:
```
import statsmodels.api as sm
Logit_model = sm.Logit(y_train_new,X_train_new)
result = Logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.363344
         Iterations 11
                            Results: Logit
================================================================================
Model:                  Logit               Method:                 MLE
Dependent Variable:     y                   Pseudo R-squared:       0.476
Date:                   2024-04-16 16:29    AIC:                    1191.7951
No. Observations:       1596                BIC:                    1277.7991
Df Model:               15                  Log-Likelihood:         -579.90
Df Residuals:           1580                LL-Null:                -1106.3
Converged:              1.0000              LLR p-value:            6.6687e-215
```

```
No. Iterations:         11.0000              Scale:                  1.0000
--------------------------------------------------------------------------------
                              Coef.   Std.Err.     z     P>|z|   [0.025   0.975]
--------------------------------------------------------------------------------
Tekanan darah (S)              0.0359   0.0120   2.9944 0.0028   0.0124  0.0595
Tekanan darah (D)             -0.0144   0.0157  -0.9173 0.3590  -0.0451  0.0163
Tinggi badan (cm)              0.0829   0.0153   5.4000 0.0000   0.0528  0.1129
Berat badan (kg)               0.0147   0.0178   0.8245 0.4096  -0.0202  0.0496
Lingkar perut (cm)            -0.0198   0.0153  -1.2968 0.1947  -0.0498  0.0101
Fat                           -0.0909   0.0256  -3.5539 0.0004  -0.1410 -0.0408
Visceral Fat                  -0.0579   0.0432  -1.3418 0.1797  -0.1425  0.0267
Masa Kerja                    -0.0296   0.0193  -1.5301 0.1260  -0.0675  0.0083
Usia_0                        -1.5710   0.2167  -7.2483 0.0000  -1.9958 -1.1462
IMT (kg/m2)_0                -10.5917   2.4616  -4.3027 0.0000 -15.4164 -5.7670
IMT (kg/m2)_1                 -7.6537   2.3511  -3.2554 0.0011 -12.2619 -3.0456
IMT (kg/m2)_2                 -8.3714   2.3504  -3.5618 0.0004 -12.9780 -3.7648
IMT (kg/m2)_3                 -8.3724   2.3082  -3.6272 0.0003 -12.8964 -3.8483
Jenis Kelamin_Pria            -2.5409   0.2339 -10.8629 0.0000  -2.9993 -2.0824
Glukosa Puasa (mg/dL)_0        0.0465   0.5317   0.0874 0.9304  -0.9957  1.0886
Trigliserida (mg/dL)_normal   -3.4499   0.2221 -15.5311 0.0000  -3.8853 -3.0146
================================================================================
```

Dapat diketahui bahwa variabel yang tidak signifikan memiliki nilai p-value di atas 0.05 yaitu variabel Tekanan darah (D), Berat badan (kg), Lingkar perut (cm), Visceral Fat, Masa Kerja, dan Glukosa Puasa (mg/dL). Selantunya dapat dilakukan eliminasi pada variabel yang tidak signifikan untuk dibuat model lagi.

Tahap 2

[ ]:
```
y = df['Cholesterol Total (mg/dL)_tinggi']
X = df.drop(['Cholesterol Total (mg/dL)_tinggi','Cholesterol Total (mg/
 ↪dL)_normal','Jenis Kelamin_Wanita','Glukosa Puasa (mg/dL)_1',
             'Trigliserida (mg/dL)_tinggi','Usia_1','Tekanan darah (D)','Berat␣
 ↪badan (kg)','Lingkar perut (cm)','Visceral Fat',
             'Masa Kerja','Glukosa Puasa (mg/dL)_0'], axis=1)
```

[ ]:
```
import statsmodels.api as sm
Logit_model = sm.Logit(y,X)
result = Logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.347779
         Iterations 7
                                Results: Logit
================================================================================
=
Model:              Logit                           Method:           MLE
```

```
Dependent Variable: Cholesterol Total (mg/dL)_tinggi Pseudo R-squared: 0.170
Date:               2024-04-16 16:29                 AIC:              951.3528
No. Observations:   1339                             BIC:              1003.3495
Df Model:           9                                Log-Likelihood:   -465.68
Df Residuals:       1329                             LL-Null:          -561.04
Converged:          1.0000                           LLR p-value:
2.8998e-36
No. Iterations:     7.0000                           Scale:            1.0000
-------------------------------------------------------------------------------
-
                                Coef.  Std.Err.    z     P>|z|    [0.025
0.975]
-------------------------------------------------------------------------------
-
Tekanan darah (S)                0.0219   0.0086   2.5570 0.0106   0.0051
0.0387
Tinggi badan (cm)                0.0351   0.0146   2.4090 0.0160   0.0065
0.0637
Fat                             -0.0191   0.0290  -0.6607 0.5088  -0.0759
0.0376
Usia_0                          -0.8026   0.2186  -3.6722 0.0002  -1.2309
-0.3742
IMT (kg/m2)_0                   -6.4826   2.6188  -2.4754 0.0133 -11.6154
-1.3499
IMT (kg/m2)_1                   -5.7741   2.6351  -2.1913 0.0284 -10.9387
-0.6095
IMT (kg/m2)_2                   -5.7737   2.6603  -2.1703 0.0300 -10.9878
-0.5596
IMT (kg/m2)_3                   -5.9589   2.6653  -2.2357 0.0254 -11.1827
-0.7350
Jenis Kelamin_Pria              -1.3015   0.2563  -5.0785 0.0000  -1.8038
-0.7992
Trigliserida (mg/dL)_normal     -2.4952   0.2248 -11.0992 0.0000  -2.9358
-2.0546
===============================================================================
=
```

Dapat diketahui bahwa variabel pada tahap 2 yang tidak signifikan memiliki nilai p-value di atas 0.05 yaitu variabel fat. Selantunya dapat dilakukan eliminasi pada variabel yang tidak signifikan untuk dibuat model lagi.

Tahap 3

[ ]:
```
y1 = df['Cholesterol Total (mg/dL)_tinggi']
X1 = df.drop(['Cholesterol Total (mg/dL)_tinggi','Cholesterol Total (mg/
↪dL)_normal','Jenis Kelamin_Wanita','Glukosa Puasa (mg/dL)_1',
```

```
            'Trigliserida (mg/dL)_tinggi','Usia_1','Tekanan darah (D)','Berat␣
↪badan (kg)','Lingkar perut (cm)','Visceral Fat',
            'Masa Kerja','Glukosa Puasa (mg/dL)_0','Fat'], axis=1)
```

[ ]:
```
import statsmodels.api as sm
Logit_model = sm.Logit(y1,X1)
result = Logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.347941
         Iterations 7
                                Results: Logit
================================================================================
=
Model:              Logit                            Method:           MLE
Dependent Variable: Cholesterol Total (mg/dL)_tinggi Pseudo R-squared: 0.170
Date:               2024-04-16 16:29                 AIC:              949.7862
No. Observations:   1339                             BIC:              996.5833
Df Model:           8                                Log-Likelihood:   -465.89
Df Residuals:       1330                             LL-Null:          -561.04
Converged:          1.0000                           LLR p-value:
7.0638e-37
No. Iterations:     7.0000                           Scale:            1.0000
--------------------------------------------------------------------------------
-
                                Coef.  Std.Err.    z     P>|z|    [0.025
0.975]
--------------------------------------------------------------------------------
-
Tekanan darah (S)               0.0215   0.0086   2.5153 0.0119   0.0048
0.0383
Tinggi badan (cm)               0.0359   0.0145   2.4710 0.0135   0.0074
0.0643
Usia_0                         -0.7988   0.2185  -3.6561 0.0003  -1.2270
-0.3706
IMT (kg/m2)_0                  -6.9934   2.5045  -2.7924 0.0052 -11.9020
-2.0847
IMT (kg/m2)_1                  -6.3890   2.4647  -2.5922 0.0095 -11.2197
-1.5582
IMT (kg/m2)_2                  -6.4093   2.4803  -2.5841 0.0098 -11.2705
-1.5480
IMT (kg/m2)_3                  -6.5953   2.4852  -2.6538 0.0080 -11.4662
-1.7244
Jenis Kelamin_Pria             -1.2563   0.2468  -5.0908 0.0000  -1.7400
-0.7726
Trigliserida (mg/dL)_normal    -2.4950   0.2248 -11.0992 0.0000  -2.9355
```

```
-2.0544
==============================================================================
=
```

[ ]:
```
# Menambahkan istilah intercept ke variabel X
X_with_intercept = sm.add_constant(X1)

# Membuat dan melatih model regresi logistik dengan istilah intercept
Logit_model = sm.Logit(y1, X_with_intercept)
result = Logit_model.fit()

# Menampilkan nilai intercept
print("Intercept:", result.params['const'])
```

```
Optimization terminated successfully.
         Current function value: 0.347941
         Iterations 7
Intercept: -5.277382395663522
```

[ ]:
```
print(result.summary2())
```

```
                                          Results: Logit
==============================================================================
==============
Model:                    Logit                                Method:
MLE
Dependent Variable:       Cholesterol Total (mg/dL)_tinggi     Pseudo R-squared:
0.170
Date:                     2024-04-16 16:49                     AIC:
949.7862
No. Observations:         1339                                 BIC:
996.5833
Df Model:                 8                                    Log-Likelihood:
-465.89
Df Residuals:             1330                                 LL-Null:
-561.04
Converged:                1.0000                               LLR p-value:
7.0638e-37
No. Iterations:           7.0000                               Scale:
1.0000
------------------------------------------------------------------------------
--------------
                              Coef.     Std.Err.      z     P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------------
const                       -5.2774 10374438.1989  -0.0000 1.0000 -20333530.5071
```

```
20333519.9523
Tekanan darah (S)            0.0215        0.0086   2.5153 0.0119         0.0048
0.0383
Tinggi badan (cm)            0.0359        0.0145   2.4710 0.0135         0.0074
0.0643
Usia_0                      -0.7988        0.2185  -3.6561 0.0003        -1.2270
-0.3706
IMT (kg/m2)_0               -1.7160 10374438.1989  -0.0000 1.0000 -20333526.9457
20333523.5137
IMT (kg/m2)_1               -1.1116 10374438.1989  -0.0000 1.0000 -20333526.3413
20333524.1181
IMT (kg/m2)_2               -1.1319 10374438.1989  -0.0000 1.0000 -20333526.3616
20333524.0978
IMT (kg/m2)_3               -1.3179 10374438.1989  -0.0000 1.0000 -20333526.5476
20333523.9118
Jenis Kelamin_Pria          -1.2563        0.2468  -5.0908 0.0000        -1.7400
-0.7726
Trigliserida (mg/dL)_normal -2.4950        0.2248 -11.0992 0.0000        -2.9355
-2.0544
================================================================================
==============
```

Dapat dilihat seluruh variabel pada model ini telah signifikan yang dilihat dari nilai p-value yang di bawah 0.05. Sehingga model yang diperoleh ini dapat dilanjutkan untuk tes akurasi.

1.7 Evaluate model performance

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3,␣
 ↪random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[ ]:
```
LogisticRegression()
```

[ ]:
```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.
  ↪format(logreg.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.87
```

Confussion matrix

[ ]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[336   4]
 [ 48  14]]
```

[ ]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       340
           1       0.78      0.23      0.35        62

    accuracy                           0.87       402
   macro avg       0.83      0.61      0.64       402
weighted avg       0.86      0.87      0.84       402
```