QUIZ PEMBELAJARAN MESIN
HEART DATASET

VINCENT MICHAEL SUTANTO
16/398531/PA/17492

PROGRAM STUDI ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2019

Bahasa Pemrograman : Python (Jupyter, Scikit-Learn, Pandas, Seaborn)
Repositori

a. Variansi Data:
https://github.com/vincentmichael089/ML-Heart/blob/master/Heart_Data_Variability.ipynb

b. Klasifikasi :
https://github.com/vincentmichael089/ML-Heart/blob/master/Heart_Classification.ipynb

c. Klastering :
https://github.com/vincentmichael089/ML-Heart/blob/master/Heart_KMeans.ipynb

Source Code : Terlampir (di akhir tugas)
Dataset : Heart.csv

1. **Buat summary dari atribut-atribut bertipe kontinyu pada data Heart.csv dengan cara menampilkan nilai-nilai count, mean, standard deviation (std), min, quartiles and max dari atribut-atribut tersebut.**

   Count:

```
In [24]: dfnew.count(axis = 0)

Out[24]: Age          297
         Sex          297
         ChestPain    297
         RestBP       297
         Chol         297
         Fbs          297
         RestECG      297
         MaxHR        297
         ExAng        297
         Oldpeak      297
         Slope        297
         Ca           297
         Thal         297
         AHD          297
         dtype: int64
```

   Mean:

```
In [25]: dfnew.mean(axis=0)

Out[25]: Age           54.542088
         Sex            0.676768
         ChestPain      0.841751
         RestBP       131.693603
         Chol         247.350168
         Fbs            0.144781
         RestECG        0.996633
         MaxHR        149.599327
         ExAng          0.326599
         Oldpeak        1.055556
         Slope          1.602694
         Ca             0.676768
         Thal           1.326599
         AHD            0.461279
         dtype: float64
```

Min:

```
In [26]: dfnew.min(axis=0)
Out[26]: Age            29.0
         Sex             0.0
         ChestPain       0.0
         RestBP         94.0
         Chol          126.0
         Fbs             0.0
         RestECG         0.0
         MaxHR          71.0
         ExAng           0.0
         Oldpeak         0.0
         Slope           1.0
         Ca              0.0
         Thal            0.0
         AHD             0.0
         dtype: float64
```

Max:

```
In [27]: dfnew.max(axis=0)
Out[27]: Age            77.0
         Sex             1.0
         ChestPain       3.0
         RestBP        200.0
         Chol          564.0
         Fbs             1.0
         RestECG         2.0
         MaxHR         202.0
         ExAng           1.0
         Oldpeak         6.2
         Slope           3.0
         Ca              3.0
         Thal            2.0
         AHD             1.0
         dtype: float64
```

Standar Deviasi:

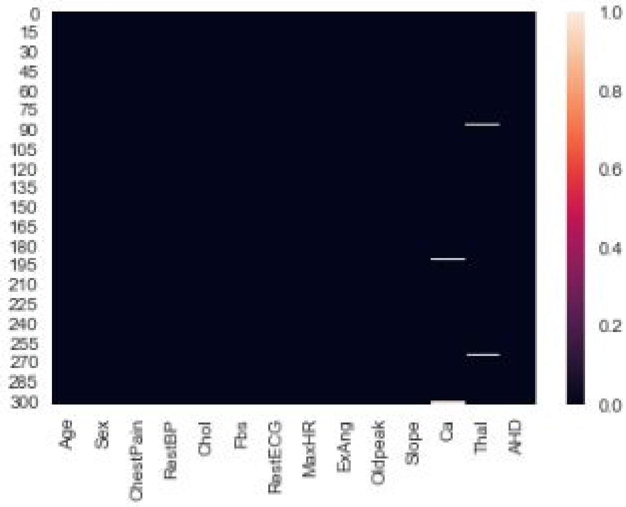
```
In [28]: dfnew.std(axis=0)
Out[28]: Age             9.049736
         Sex             0.468500
         ChestPain       0.964859
         RestBP         17.762806
         Chol           51.997583
         Fbs             0.352474
         RestECG         0.994914
         MaxHR          22.941562
         ExAng           0.469761
         Oldpeak         1.166123
         Slope           0.618187
         Ca              0.938965
         Thal            0.585061
         AHD             0.499340
         dtype: float64
```

Quartil:

```
In [31]: dfnew.quantile([0.25, 0.75], interpolation='nearest')
Out[31]:
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0.25	48.0	0.0	0.0	120.0	211.0	0.0	0.0	133.0	0.0	0.0	1.0	0.0	1.0	0.0
0.75	61.0	1.0	1.0	140.0	276.0	0.0	2.0	166.0	1.0	1.6	2.0	1.0	2.0	1.0

2. **Tentukan jumlah missing value dari masing-masing atribut. Replace semua missing value.**



Dari Heatmap tersebut terlihat bahwa hanya ada 2 atribut yang memiliki *missing value* yaitu atribut ‘Ca’ dan atribut ‘Thal’.

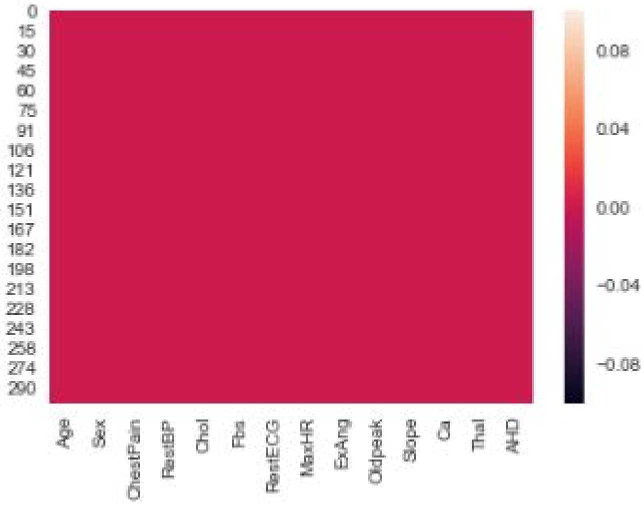
In [8]:
```
print(dfnew['Ca'].describe(),"\n")
print(dfnew['Thal'].describe())
print("\ndata missing dari Ca = 303 - 299 = ",303-299)
print("data missing dari Thal = 303 - 301 = ",303-301)
```

```
count     299
unique      4
top         0
freq      176
Name: Ca, dtype: object

count        301
unique         3
top       normal
freq         166
Name: Thal, dtype: object

data missing dari Ca = 303 - 299 =  4
data missing dari Thal = 303 - 301 =  2
```

Kita *describe* kolom pada *data frame* tersebut maka dapat terlihat data missing dari atribut ‘Ca’ berjumlah 4 dan data missing dari atribut ‘Thal’ berjumlah 2 data. Data yang hilang tersebut kita hapus saja *row*-nya dari dataset karena jumlahnya sedikit sehingga dapat dianggap sebagai *outlier data* saja.

In [9]:
```
dfnew = dfnew.dropna()
missing_values = dfnew.isnull()

sns.heatmap(data = missing_values)
dfnew.describe()
```

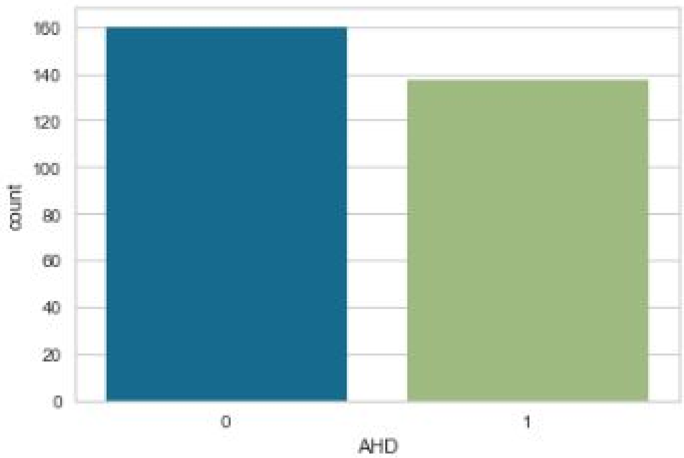
Heatmap berikut menunjukan bahwa tidak ada lagi data yang memiliki *missing value.*



3. Atribut klas dari data ini adalah atribut AHD. Tentukan jumlah data untuk masing-masing klas (klas YES dan NO).

```
In [12]: sns.countplot(x='AHD', data=dfnew)
         dfnew.AHD.value_counts()
```

```
0    160
1    137
Name: AHD, dtype: int64
```

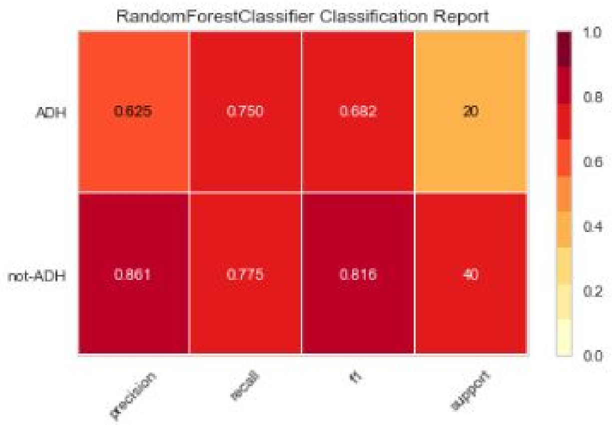


Atribut klas AHD memiliki 160 klas NO dan 137 klas YES (setelah dilakukan penghilangan data bernilai null)

4. **Menggunakan k-cross validation, dimana k = 5, tentukan nilai accuracy, Precision, Recall dari model yang dibuat menggunakan algoritma Random Forest, AdaBoost, dan Gradient Boosting**

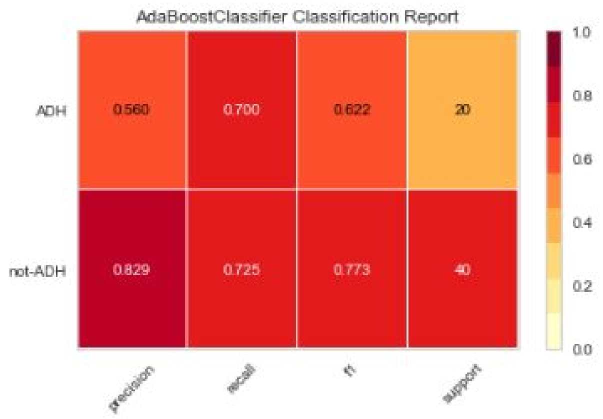
Random Forest

```
              precision    recall  f1-score   support

     not-ADH       0.83      0.75      0.79        40
         ADH       0.58      0.70      0.64        20

   micro avg       0.73      0.73      0.73        60
   macro avg       0.71      0.72      0.71        60
weighted avg       0.75      0.73      0.74        60
```



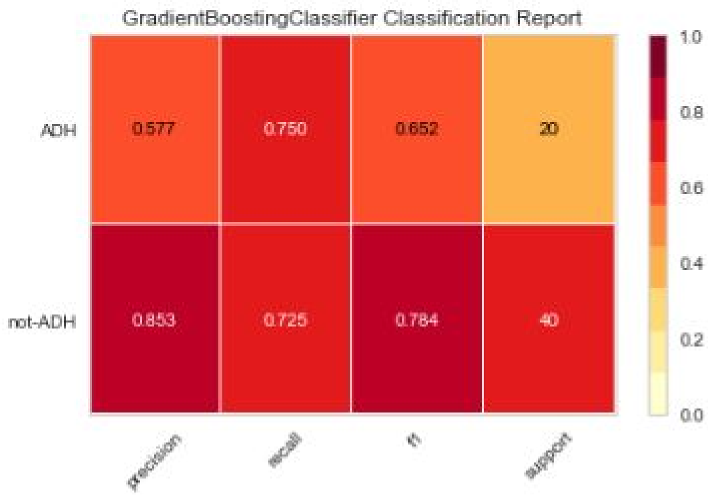
AdaBoost

```
              precision    recall  f1-score   support

     not-ADH       0.83      0.72      0.77        40
         ADH       0.56      0.70      0.62        20

   micro avg       0.72      0.72      0.72        60
   macro avg       0.69      0.71      0.70        60
weighted avg       0.74      0.72      0.72        60
```



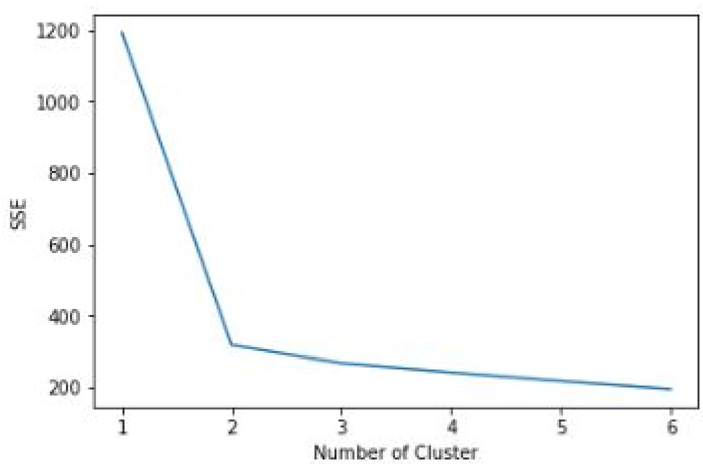
Gradient Boosting

```
              precision    recall  f1-score   support

     not-ADH       0.85      0.72      0.78        40
         ADH       0.58      0.75      0.65        20

   micro avg       0.73      0.73      0.73        60
   macro avg       0.71      0.74      0.72        60
weighted avg       0.76      0.73      0.74        60
```



5. **Tentukan nilai K terbaik jika anda melakukan klastering data Heart.csv menggunakan algoritma K-means dengan mengambil nilai k = 2, 3, 4, 5, 6. Buat plot untuk nilai SSE (sumbu-Y) terhadap nilai K (sumbu-X).**

   Dengan menggunakan Elbow Criterion Model dapat terlihat bahwa jumlah cluster terbaik untuk merepresentasikan data didapat ketika K = 2 (Ditunjukan pada diagram berikut)



Variabilitas data dari dataset Heart

1. Inisialisasikan library yang diperlukan untuk dataset ini.

In [1]:
```
import numpy as np
import pandas as pd

from sklearn.preprocessing import LabelEncoder
```

2. Masukan dataset Heart.csv kedalam dataframe

2.1 Siapkan dataframe untuk mengambil attribut

In [2]:
```
df = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\heartbeat dataset\\Heart.csv",h
eader=None, skipinitialspace=True)
df = df.drop(df.columns[0], axis=1)

df.head()
```

Out[2]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	typical	145	233	1	2	150	0	2.3	3	0	fixed	No
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3	normal	Yes
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2	reversable	Yes
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0	normal	No

2.2 Masukan attribut kedalam array

In [3]:
```
attrs = []
for attr in range(1,15):
    attrs.append(df.at[0,attr])
attrs
```

Out[3]:
```
['Age',
 'Sex',
 'ChestPain',
 'RestBP',
 'Chol',
 'Fbs',
 'RestECG',
 'MaxHR',
 'ExAng',
 'Oldpeak',
 'Slope',
 'Ca',
 'Thal',
 'AHD']
```

2.3 Dataframe baru dengan nama kolom = attrs, tampilkan dataframe yang dihasilkan

In [4]:
```
dfnew = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\heartbeat dataset\\Heart.cs
v",header=None, skipinitialspace=True)
dfnew = df.iloc[1:]
dfnew.drop(dfnew.index[[0,1]])
dfnew.columns = attrs
dfnew.index =  range(len(dfnew.index))
print(dfnew.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
Age          303 non-null object
Sex          303 non-null object
ChestPain    303 non-null object
RestBP       303 non-null object
Chol         303 non-null object
Fbs          303 non-null object
RestECG      303 non-null object
MaxHR        303 non-null object
ExAng        303 non-null object
Oldpeak      303 non-null object
Slope        303 non-null object
Ca           299 non-null object
Thal         301 non-null object
AHD          303 non-null object
dtypes: object(14)
memory usage: 33.2+ KB
None
```

3. Preprocessing

In [5]:
```
dfnew = dfnew.dropna()
missing_values = dfnew.isnull()

print("ChestPain :\n",dfnew['ChestPain'].unique().tolist(),"\n")
print("Thal :\n",dfnew['Thal'].unique().tolist(),"\n")

dfnew.describe()
```

```
ChestPain :
 ['typical', 'asymptomatic', 'nonanginal', 'nontypical']

Thal :
 ['fixed', 'normal', 'reversable']
```

Out[5]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
count	297	297	297	297	297	297	297	297	297	297	297	297	297	297
unique	41	2	4	50	152	2	3	91	2	40	3	4	3	2
top	58	1	asymptomatic	120	234	0	0	162	0	0	1	0	normal	No
freq	18	201	142	37	6	254	147	11	200	96	139	174	164	160

In [6]:
```
lb = LabelEncoder()
dfnew['AHD'] = lb.fit_transform(dfnew['AHD'])
dfnew['ChestPain'] = lb.fit_transform(dfnew['ChestPain'])
dfnew['Thal'] = lb.fit_transform(dfnew['Thal'])
dfnew.head()
```

Out[6]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	63	1	3	145	233	1	2	150	0	2.3	3	0	0	0
1	67	1	0	160	286	0	2	108	1	1.5	2	3	1	1
2	67	1	0	120	229	0	2	129	1	2.6	2	2	2	1
3	37	1	1	130	250	0	0	187	0	3.5	3	0	1	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0	1	0

4. Summary dari Atribut Kontinu

In [7]:
```
dfnew = dfnew.astype('float64')
```

In [8]:
```
dfnew.count(axis = 0)
```

Out[8]:
```
Age          297
Sex          297
ChestPain    297
RestBP       297
Chol         297
Fbs          297
RestECG      297
MaxHR        297
ExAng        297
Oldpeak      297
Slope        297
Ca           297
Thal         297
AHD          297
dtype: int64
```

In [9]:
```
dfnew.mean(axis=0)
```

Out[9]:
```
Age           54.542088
Sex            0.676768
ChestPain      0.841751
RestBP       131.693603
Chol         247.350168
Fbs            0.144781
RestECG        0.996633
MaxHR        149.599327
ExAng          0.326599
Oldpeak        1.055556
Slope          1.602694
Ca             0.676768
Thal           1.326599
AHD            0.461279
dtype: float64
```

In [10]:
```
dfnew.min(axis=0)
```

Out[10]:
```
Age           29.0
Sex            0.0
ChestPain      0.0
RestBP        94.0
Chol         126.0
Fbs            0.0
RestECG        0.0
MaxHR         71.0
ExAng          0.0
Oldpeak        0.0
Slope          1.0
Ca             0.0
Thal           0.0
AHD            0.0
dtype: float64
```

In [11]:
```
dfnew.max(axis=0)
```

Out[11]:
```
Age           77.0
Sex            1.0
ChestPain      3.0
RestBP       200.0
Chol         564.0
Fbs            1.0
RestECG        2.0
MaxHR        202.0
ExAng          1.0
Oldpeak        6.2
Slope          3.0
Ca             3.0
Thal           2.0
AHD            1.0
dtype: float64
```

In [12]:
```
dfnew.std(axis=0)
```

Out[12]:
```
Age           9.049736
Sex           0.468500
ChestPain     0.964859
RestBP       17.762806
Chol         51.997583
Fbs           0.352474
RestECG       0.994914
MaxHR        22.941562
ExAng         0.469761
Oldpeak       1.166123
Slope         0.618187
Ca            0.938965
Thal          0.585061
AHD           0.499340
dtype: float64
```

In [13]:
```
dfnew.quantile([0.25, 0.75], interpolation='nearest')
```

Out[13]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0.25	48.0	0.0	0.0	120.0	211.0	0.0	0.0	133.0	0.0	0.0	1.0	0.0	1.0	0.0
0.75	61.0	1.0	1.0	140.0	276.0	0.0	2.0	166.0	1.0	1.6	2.0	1.0	2.0	1.0

In [ ]:

Klasifikasi dataset Heart

1. Inisialisasikan library yang diperlukan untuk dataset ini.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder

from yellowbrick.classifier import ClassificationReport
```

2. Masukan dataset Heart.csv kedalam dataframe

2.1 Siapkan dataframe untuk mengambil attribut

In [2]:
```
df = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\heartbeat dataset\\Heart.csv",h
eader=None, skipinitialspace=True)
df = df.drop(df.columns[0], axis=1)

df.head()
```

Out[2]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	typical	145	233	1	2	150	0	2.3	3	0	fixed	No
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3	normal	Yes
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2	reversable	Yes
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0	normal	No

2.2 Masukan attribut kedalam array

In [3]:
```
attrs = []
for attr in range(1,15):
    attrs.append(df.at[0,attr])
attrs
```

Out[3]:
```
['Age',
 'Sex',
 'ChestPain',
 'RestBP',
 'Chol',
 'Fbs',
 'RestECG',
 'MaxHR',
 'ExAng',
 'Oldpeak',
 'Slope',
 'Ca',
 'Thal',
 'AHD']
```

2.3 Dataframe baru dengan nama kolom = attrs, tampilkan dataframe yang dihasilkan

In [4]:
```
dfnew = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\heartbeat dataset\\Heart.cs
v",header=None, skipinitialspace=True)
dfnew = df.iloc[1:]
dfnew.drop(dfnew.index[[0,1]])
dfnew.columns = attrs
dfnew.index =  range(len(dfnew.index))
print(dfnew.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
Age          303 non-null object
Sex          303 non-null object
ChestPain    303 non-null object
RestBP       303 non-null object
Chol         303 non-null object
Fbs          303 non-null object
RestECG      303 non-null object
MaxHR        303 non-null object
ExAng        303 non-null object
Oldpeak      303 non-null object
Slope        303 non-null object
Ca           299 non-null object
Thal         301 non-null object
AHD          303 non-null object
dtypes: object(14)
memory usage: 33.2+ KB
None
```

In [5]:
```
print(dfnew.describe())
```

```
        Age  Sex     ChestPain RestBP Chol  Fbs RestECG MaxHR ExAng Oldpeak  \
count   303  303           303    303  303  303     303   303   303     303
unique   41    2             4     50  152    2       3    91     2      40
top      58    1  asymptomatic    120  234    0       0   162     0       0
freq     19  206           144     37    6  258     151    11   204      99

       Slope   Ca    Thal  AHD
count    303  299     301  303
unique     3    4       3    2
top        1    0  normal   No
freq     142  176     166  164
```

In [6]:
```
dfnew.head()
```

Out[6]:

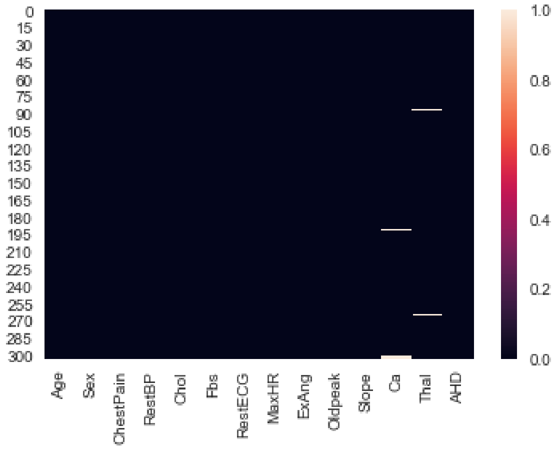
	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	63	1	typical	145	233	1	2	150	0	2.3	3	0	fixed	No
1	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3	normal	Yes
2	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2	reversable	Yes
3	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0	normal	No
4	41	0	nontypical	130	204	0	2	172	0	1.4	1	0	normal	No

3. Cek data null

3.1 Cek apakah ada missing value

In [7]:
```
missing_values = dfnew.isnull()

sns.heatmap(data = missing_values)
```

Out[7]: `<matplotlib.axes._subplots.AxesSubplot at 0x2623e7f9898>`



3.2 Dari heatmap terpancar bahwa kolom Ca dan Thal memiliki missing value

describe df untuk mengetahui jumlah missing value tiap kolom

In [8]:
```
print(dfnew['Ca'].describe(),"\n")
print(dfnew['Thal'].describe())
print("\ndata missing dari Ca = 303 - 299 = ",303-299)
print("data missing dari Thal = 303 - 301 = ",303-301)
```

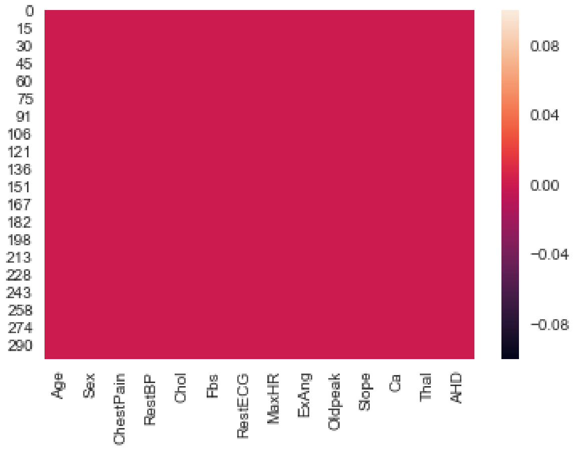
```
count     299
unique      4
top         0
freq      176
Name: Ca, dtype: object

count        301
unique         3
top       normal
freq         166
Name: Thal, dtype: object

data missing dari Ca = 303 - 299 =  4
data missing dari Thal = 303 - 301 =  2
```

3.3 Hilangkan kolom yang memiliki missing value

In [9]:
```
dfnew = dfnew.dropna()
missing_values = dfnew.isnull()

sns.heatmap(data = missing_values)
dfnew.describe()
```

Out[9]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
count	297	297	297	297	297	297	297	297	297	297	297	297	297	297
unique	41	2	4	50	152	2	3	91	2	40	3	4	3	2
top	58	1	asymptomatic	120	197	0	0	162	0	0	1	0	normal	No
freq	18	201	142	37	6	254	147	11	200	96	139	174	164	160



4. Representasikan data 'non-numerik' kedalam 'numerik'

4.1 List isi dari kolom ChestPain dan Thal

In [10]:
```
#for attr in attrs:
#   print(attr," :\n",dfnew[attr].unique().tolist(),"\n")

print("ChestPain :\n",dfnew['ChestPain'].unique().tolist(),"\n")
print("Thal :\n",dfnew['Thal'].unique().tolist(),"\n")
```

```
ChestPain :
 ['typical', 'asymptomatic', 'nonanginal', 'nontypical']

Thal :
 ['fixed', 'normal', 'reversable']
```

4.2 Label Encoder kolom ChestPain, Thal, AHD

In [11]:
```
lb = LabelEncoder()
dfnew['AHD'] = lb.fit_transform(dfnew['AHD'])
dfnew['ChestPain'] = lb.fit_transform(dfnew['ChestPain'])
dfnew['Thal'] = lb.fit_transform(dfnew['Thal'])
dfnew.head()
```

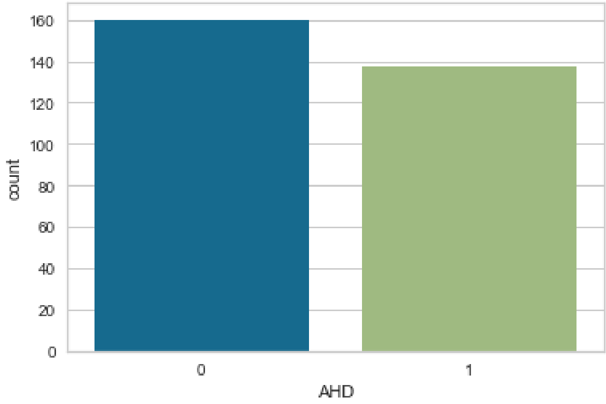
Out[11]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	63	1	3	145	233	1	2	150	0	2.3	3	0	0	0
1	67	1	0	160	286	0	2	108	1	1.5	2	3	1	1
2	67	1	0	120	229	0	2	129	1	2.6	2	2	2	1
3	37	1	1	130	250	0	0	187	0	3.5	3	0	1	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0	1	0

5. Lihat perbandingan jumlah data 0 dan 1 pada target (AHD)

In [12]:
```
sns.countplot(x='AHD', data=dfnew)
dfnew.AHD.value_counts()
```

Out[12]:
```
0    160
1    137
Name: AHD, dtype: int64
```



6. Normalisasi Data

In [13]:
```
feature = attrs
feature.pop()
feature
```

Out[13]:
```
['Age',
 'Sex',
 'ChestPain',
 'RestBP',
 'Chol',
 'Fbs',
 'RestECG',
 'MaxHR',
 'ExAng',
 'Oldpeak',
 'Slope',
 'Ca',
 'Thal']
```

In [14]:
```
features = dfnew[feature]
label = dfnew['AHD']
```

In [15]:
```
scaler = MinMaxScaler(feature_range = (0,1))
col = features.columns.tolist()

normalised_feature = features
normalised_feature[col] = scaler.fit_transform(normalised_feature[col])

normalised_feature.head()
```

```
C:\Users\aftermath\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:323: DataConversionWa
rning: Data with input dtype int32, object were all converted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
C:\Users\aftermath\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#in
dexing-view-versus-copy
  """
C:\Users\aftermath\Anaconda3\lib\site-packages\pandas\core\indexing.py:543: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#in
dexing-view-versus-copy
  self.obj[item] = s
```

Out[15]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
0	0.708333	1.0	1.000000	0.481132	0.244292	1.0	1.0	0.603053	0.0	0.370968	1.0	0.000000	0.0
1	0.791667	1.0	0.000000	0.622642	0.365297	0.0	1.0	0.282443	1.0	0.241935	0.5	1.000000	0.5
2	0.791667	1.0	0.000000	0.245283	0.235160	0.0	1.0	0.442748	1.0	0.419355	0.5	0.666667	1.0
3	0.166667	1.0	0.333333	0.339623	0.283105	0.0	0.0	0.885496	0.0	0.564516	1.0	0.000000	0.5
4	0.250000	0.0	0.666667	0.339623	0.178082	0.0	1.0	0.770992	0.0	0.225806	0.0	0.000000	0.5

7. Training dan Validasi dengan K-Fold (5 Fold)

7.1 Random Forest

In [16]:
```
rfmodel = RandomForestClassifier(n_estimators=190, criterion='gini', n_jobs=-1)
score =  cross_val_score(rfmodel, normalised_feature, label, cv=5)
print("score: ",score.mean())
```

```
score:  0.8211864406779661
```

7.2 AdaBoost

In [17]:
```
adaboostmodel = AdaBoostClassifier(n_estimators=25, random_state=101)
score =  cross_val_score(adaboostmodel, normalised_feature, label, cv=5)
print("score: ",score.mean())
```

```
score:  0.8110169491525424
```

7.3 Gradient Boosting

In [18]:
```
gbmodel = GradientBoostingClassifier(n_estimators=25, random_state=101)
score =  cross_val_score(gbmodel, normalised_feature, label, cv=5)
print("score: ",score.mean())
```

```
score:  0.8077966101694913
```

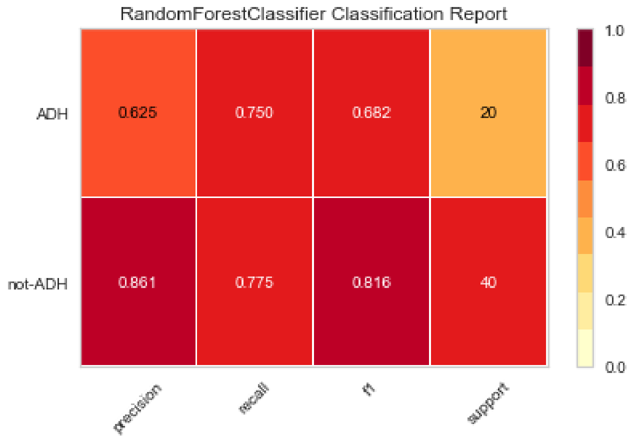
8. Testing

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(normalised_feature, label, test_size=0.20, ran
dom_state=101)
classes = ['not-ADH','ADH']
```

8.1 Random Forest

In [20]:
```
rfmodel.fit(X_train, y_train)
y_pred = rfmodel.predict(X_test)
print(classification_report(y_test, y_pred, target_names = classes))

visualizer = ClassificationReport(rfmodel, classes=classes, support=True)
visualizer.fit(X_train, y_train)  # Fit the visualizer and the model
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
g = visualizer.poof()             # Draw/show/poof the data
```

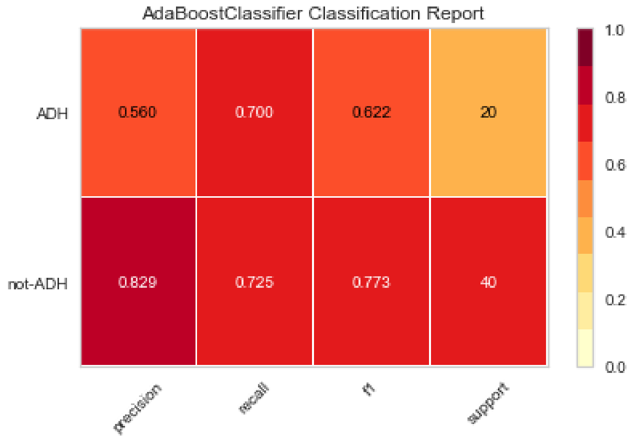
```
              precision    recall  f1-score   support

     not-ADH       0.83      0.75      0.79        40
         ADH       0.58      0.70      0.64        20

   micro avg       0.73      0.73      0.73        60
   macro avg       0.71      0.72      0.71        60
weighted avg       0.75      0.73      0.74        60
```



8.2 AdaBoost

In [21]:
```
adaboostmodel.fit(X_train, y_train)
y_pred = adaboostmodel.predict(X_test)
print(classification_report(y_test, y_pred, target_names = classes))

visualizer = ClassificationReport(adaboostmodel, classes=classes, support=True)
visualizer.fit(X_train, y_train)  # Fit the visualizer and the model
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
g = visualizer.poof()             # Draw/show/poof the data
```

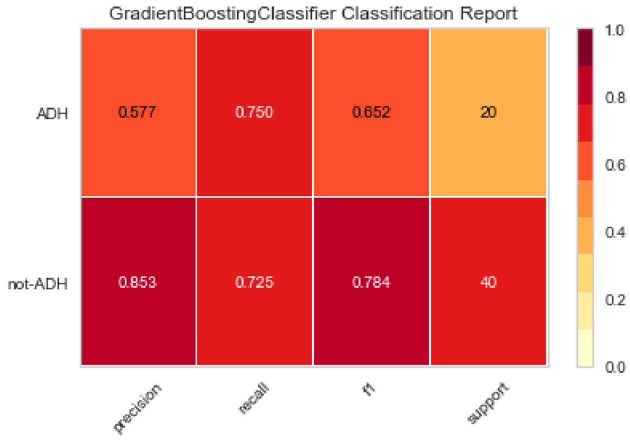
```
              precision    recall  f1-score   support

     not-ADH       0.83      0.72      0.77        40
         ADH       0.56      0.70      0.62        20

   micro avg       0.72      0.72      0.72        60
   macro avg       0.69      0.71      0.70        60
weighted avg       0.74      0.72      0.72        60
```



8.3 Gradient Boosting

In [22]:
```
gbmodel.fit(X_train, y_train)
y_pred = gbmodel.predict(X_test)
print(classification_report(y_test, y_pred, target_names = classes))

visualizer = ClassificationReport(gbmodel, classes=classes, support=True)
visualizer.fit(X_train, y_train)  # Fit the visualizer and the model
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
g = visualizer.poof()             # Draw/show/poof the data
```

```
              precision    recall  f1-score   support

     not-ADH       0.85      0.72      0.78        40
         ADH       0.58      0.75      0.65        20

   micro avg       0.73      0.73      0.73        60
   macro avg       0.71      0.74      0.72        60
weighted avg       0.76      0.73      0.74        60
```



In [ ]:

K-Means Clustering pada dataset Heart

1. Inisialisasikan library yang diperlukan untuk dataset ini.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_score
```

2. Masukan dataset Heart.csv kedalam dataframe

In [2]:
```
df = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\heartbeat dataset\\Heart.csv",h
eader=None, skipinitialspace=True)
df = df.drop(df.columns[0], axis=1)

df.head()
```

Out[2]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	typical	145	233	1	2	150	0	2.3	3	0	fixed	No
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3	normal	Yes
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2	reversable	Yes
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0	normal	No

In [3]:
```
attrs = []
for attr in range(1,15):
    attrs.append(df.at[0,attr])
attrs
```

Out[3]:
```
['Age',
 'Sex',
 'ChestPain',
 'RestBP',
 'Chol',
 'Fbs',
 'RestECG',
 'MaxHR',
 'ExAng',
 'Oldpeak',
 'Slope',
 'Ca',
 'Thal',
 'AHD']
```

In [4]:
```
dfnew = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\heartbeat dataset\\Heart.cs
v",header=None, skipinitialspace=True)
dfnew = df.iloc[1:]
dfnew.drop(dfnew.index[[0,1]])
dfnew.columns = attrs
dfnew.index =  range(len(dfnew.index))
print(dfnew.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
Age          303 non-null object
Sex          303 non-null object
ChestPain    303 non-null object
RestBP       303 non-null object
Chol         303 non-null object
Fbs          303 non-null object
RestECG      303 non-null object
MaxHR        303 non-null object
ExAng        303 non-null object
Oldpeak      303 non-null object
Slope        303 non-null object
Ca           299 non-null object
Thal         301 non-null object
AHD          303 non-null object
dtypes: object(14)
memory usage: 33.2+ KB
None
```

3. Drop missing value dan ubah nilai non-numerik kedalam numerik

In [5]:
```
dfnew = dfnew.dropna()
missing_values = dfnew.isnull()

print("ChestPain :\n",dfnew['ChestPain'].unique().tolist(),"\n")
print("Thal :\n",dfnew['Thal'].unique().tolist(),"\n")

dfnew.describe()
```

```
ChestPain :
 ['typical', 'asymptomatic', 'nonanginal', 'nontypical']

Thal :
 ['fixed', 'normal', 'reversable']
```

Out[5]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
count	297	297	297	297	297	297	297	297	297	297	297	297	297	297
unique	41	2	4	50	152	2	3	91	2	40	3	4	3	2
top	58	1	asymptomatic	120	234	0	0	162	0	0	1	0	normal	No
freq	18	201	142	37	6	254	147	11	200	96	139	174	164	160

In [6]:
```
lb = LabelEncoder()
dfnew['AHD'] = lb.fit_transform(dfnew['AHD'])
dfnew['ChestPain'] = lb.fit_transform(dfnew['ChestPain'])
dfnew['Thal'] = lb.fit_transform(dfnew['Thal'])
dfnew.head()
```

Out[6]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	63	1	3	145	233	1	2	150	0	2.3	3	0	0	0
1	67	1	0	160	286	0	2	108	1	1.5	2	3	1	1
2	67	1	0	120	229	0	2	129	1	2.6	2	2	2	1
3	37	1	1	130	250	0	0	187	0	3.5	3	0	1	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0	1	0

In [7]:
```
feature = attrs
feature.pop()
feature
```

Out[7]:
```
['Age',
 'Sex',
 'ChestPain',
 'RestBP',
 'Chol',
 'Fbs',
 'RestECG',
 'MaxHR',
 'ExAng',
 'Oldpeak',
 'Slope',
 'Ca',
 'Thal']
```

In [8]:
```
features = dfnew[feature]
label = dfnew['AHD']
```

4. Normalisasi data

In [9]:
```
scaler = MinMaxScaler(feature_range = (0,1))
col = features.columns.tolist()

normalised_feature = features
normalised_feature[col] = scaler.fit_transform(normalised_feature[col])

normalised_feature.head()
```

```
C:\Users\aftermath\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:323: DataConversionWa
rning: Data with input dtype int32, object were all converted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
C:\Users\aftermath\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#in
dexing-view-versus-copy
  """
C:\Users\aftermath\Anaconda3\lib\site-packages\pandas\core\indexing.py:543: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#in
dexing-view-versus-copy
  self.obj[item] = s
```

Out[9]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
0	0.708333	1.0	1.000000	0.481132	0.244292	1.0	1.0	0.603053	0.0	0.370968	1.0	0.000000	0.0
1	0.791667	1.0	0.000000	0.622642	0.365297	0.0	1.0	0.282443	1.0	0.241935	0.5	1.000000	0.5
2	0.791667	1.0	0.000000	0.245283	0.235160	0.0	1.0	0.442748	1.0	0.419355	0.5	0.666667	1.0
3	0.166667	1.0	0.333333	0.339623	0.283105	0.0	0.0	0.885496	0.0	0.564516	1.0	0.000000	0.5
4	0.250000	0.0	0.666667	0.339623	0.178082	0.0	1.0	0.770992	0.0	0.225806	0.0	0.000000	0.5

5. Clustering

In [10]:
```
kmeans = KMeans(n_clusters=2)
kmeans.fit(normalised_feature)
```

Out[10]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [11]:
```
kmeans = KMeans(n_clusters=3)
kmeans.fit(normalised_feature)
```

Out[11]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [12]:
```
kmeans = KMeans(n_clusters=4)
kmeans.fit(normalised_feature)
```

Out[12]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [13]:
```
kmeans = KMeans(n_clusters=5)
kmeans.fit(normalised_feature)
```

Out[13]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

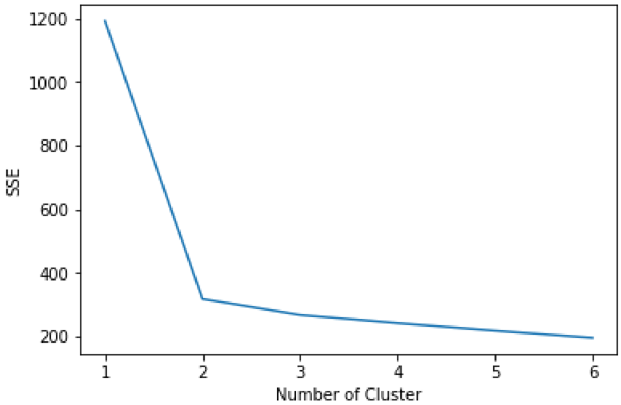
In [14]:
```
kmeans = KMeans(n_clusters=6)
kmeans.fit(normalised_feature)
```

Out[14]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=6, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

6. Evaluasi K-Means

6.1. Elbow Criterion Method

In [17]:
```
sse = {}
for k in range(1, 7):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(normalised_feature)
    normalised_feature["clusters"] = kmeans.labels_
    #print(data["clusters"])
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of samples to their closest cluster cente
r
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of Cluster")
plt.ylabel("SSE")
plt.show()
```



6.2. Silhouette Coefficient Method

In [18]:
```
for n_cluster in range(2, 7):
    kmeans = KMeans(n_clusters=n_cluster).fit(normalised_feature)
    label_ = kmeans.labels_
    sil_coeff = silhouette_score(normalised_feature, label_, metric='euclidean')
    print("For n_clusters={}, The Silhouette Coefficient is {}".format(n_cluster, sil_coeff))
```

```
For n_clusters=2, The Silhouette Coefficient is 0.47132472284927796
For n_clusters=3, The Silhouette Coefficient is 0.3968807120538654
For n_clusters=4, The Silhouette Coefficient is 0.35904993799842577
For n_clusters=5, The Silhouette Coefficient is 0.3898274950799745
For n_clusters=6, The Silhouette Coefficient is 0.4225011768185195
```

In [ ]: