LAPORAN RESPONSI PRAKTIKUM

DATA MINING

(Analisis Menggunakan Metode *K-means Clustering* pada Terapi kulit dengan *Data Cryotherapy*)

Disusun oleh :

Yuly Alfiani

1800018308

Slot Senin 13.30

Kelas A

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS AHMAD DAHLAN YOGYAKARTA

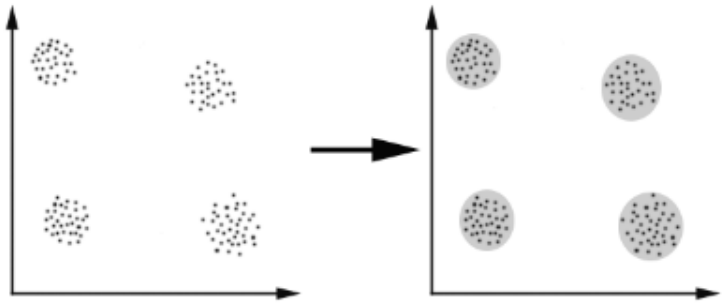
2021/2022

Daftar Isi

1. Latar Belakang

K-means Clustering adalah salah satu metode data *clustering non hirarki* yang berusaha mempartisi data yang ada ke dalam satu atau lebih cluster atau kelompok. Metode ini mempratisi data yang ada dalam satu cluster atau kelompok sehingga data yang memiliki karakteristik yang sama dikelompokkan ke dalam satu cluster yang sama dan data yang mempunyai karakteristik yang berbeda dikelompokkan ke dalam kelompok lain.

Berikut gambaran konsep kerja dari proses algoritma Kmeans clustering



Pada gambar diatas kita dapat dengan mudah mengidentifikasikan 4 kelompok menjadi data yang dapat dibagi yaitu kesamaan dengan kriteria jarak antara dua atau lebih benda dalam klister yang sama jika mereka dekat dan sesuai dengan jarak yang diberikan,hal ini disebut distance-based clustering.

Cryotherapy adalah terapi dimana permukaan kulit atau mukosa dibekukan dengan zat yang disebut *cryogens. Cryogens* yang dipakai dalam prosedur ini meliputi nitrogen cair yang sekarang paling banyak dipakai , carbon dioxide snow, serta dimenthyl ether dan propane. Alasan mengambil topik ini adalah untuk mengetahui dan menganalisis seberapa penting perawatan kullit dengan perawatan *cryotherapy* dengan metode *K-means clustering* karna dengan metode tersebut bisa memprediksi peluang atau hasil yang akurat. serta hasil yang ingin dicapai pada metode *K-means clustering* adalah pada data tersebut menghasilkan data prediksi atau hasil yang akurat bahwa perawatan menggunakan *cryotherapy* sangat penting bagi perawatan kulit atau bagi seseorang yang sedang terkena penyakit kulit sehingga menggunakan perawatan *cryotherapy.*

2. Tujuan

Tujuan dari analisis ini adalah untuk mendapatkan data prediksi atau hasil yang akurat dari data set Cryotherapy bahwa perawatan menggunakan cryotherapy sangat penting bagi perawatan atau bagi seseorang yang sedang terkena penyakit kulit sehingga menggunakan perawatan cryotherapy.

3. Dataset

Pembagian data dilakukan menggunakan Train Test Split untuk membagi dataset menjadi data training dan data testing . pembagiannya yaitu data training 80 % dan data testing 20%, yang artinya dari 630 data, data training berisi 504 data dan data testing berisi 120 data. Setelah dilakukan pemisahan , selanjutnya akan dilakukan prediksi set dan test set, sampai pada perhitungan nilai akurasi untuk mendapatkan hasil.

4. Manfaat Studi Kasus

Manfaat dari studi kasus adalah dapat mengetahui perhitungan menggunakan K-Means Clustering dengan menggunakan data set Cryotheraphy bahwa perawatan menggunakan cryotherapy sangan penting bagi perawatan kulit sehingga mengggunakan perawatan cryotherapy.

Adapun kekurangan dan kelebihan menggunakan metode K-Menas Clustering adalah sebagai berikut:

Kelebihan dari metode *K-means clustering* :

- Mudah diimplementasikan dan dijalankan
- Umum digunakan.

Kekurangan dari metode *K-means clustering* :

- Sebelum algoritma dijalankan, k titik diinisialisasi secara random sehingga pengelompokkan data yang dihasilkan dapat berbeda-beda. Jika nilai random untuk inilisialisasi kurang baik maka pengelompokkan yang dihasilkanpun kurang menjadi optimal.

5. Langkah – Langkah

a. Perhitungan Manual

- Cleaning

Dalam proses cleaning didalam data set cryotherapy tidak ditemukan data yang harus dibersihkan karena didalam dataset cryotherapy tidak ada deret yang kosong, sudah dibuktikan melalui python menggunakan fungsi ".empty" dengan outputnya false (tidak terdapat deret data yang kosong dalam data set)

sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment	Total
1	35	12	5	1	100	0	154
1	29	7	5	1	96	1	140
1	50	8	1	3	132	0	195
1	32	11,75	7	3	750	0	804,75
1	67	9,25	1	1	42	0	121,25
1	41	8	2	2	20	1	75
1	36	11	2	1	8	0	59
1	59	3,5	3	3	20	0	89,5
1	20	4,5	12	1	6	1	45,5
2	34	11,25	3	3	150	0	203,25
						Jumlah	1887,25
						Rata-rata	188,725

- Integration data

Proses integratioin data menggunakan metode RFM yaitu Recency, Frequency dan Monetary , perhitungannya sebagai berikut :

Recency = 96/3 = 32

Frequency = 42/12 = 3.5

Monetary = 750/10 = 75

- Metode Kmeans Clustering

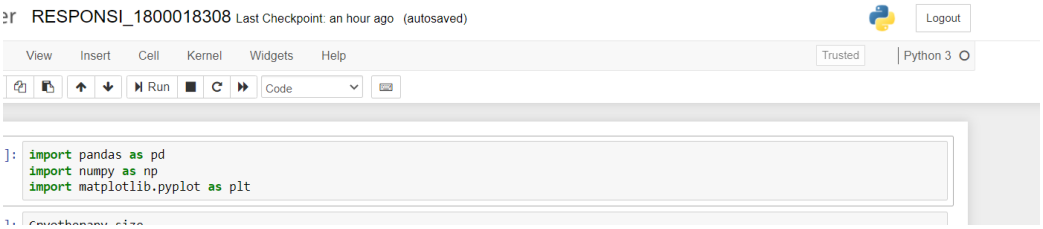
sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment
1	35	12	5	1	100	0
1	29	7	5	1	96	1
1	50	8	1	3	132	0
1	32	11,75	7	3	750	0
1	67	9,25	1	1	42	0
1	41	8	2	2	20	1
1	36	11	2	1	8	0
1	59	3,5	3	3	20	0
1	20	4,5	12	1	6	1
2	34	11,25	3	3	150	0

- $\sqrt{(35-5)^2+(12-5)^2+(100-5)^2}$ = $\sqrt{60+49+9.025}$ = $\sqrt{9.134}$= 95.571
- $\sqrt{(29-5)^2+(7-5)^2+(96-5)^2}$ = $\sqrt{576+4+8.281}$ = $\sqrt{8861}$= 94.13
- $\sqrt{(50-1)^2+(8-1)^2+(132-1)^2}$ = $\sqrt{2.401+4+9.025}$ = $\sqrt{11.430}$= 106.911
- $\sqrt{(35-7)^2+(11{,}75-7)^2+(750-7)^2}$ = $\sqrt{784+22.56+552.049}$ = $\sqrt{552.855}$=743.54
- $\sqrt{(67-1)^2+(9.24-1)^2+(42-1)^2}$ = $\sqrt{4.356+85.37+1.681}$ = $\sqrt{1.770}$= 42.07

- $\sqrt{(41-2)^2+(8-2)^2+(20-2)^2}$ = $\sqrt{1.521+14+324}$ = $\sqrt{1.859}$= 43.11
- $\sqrt{(36-2)^2+(11-2)^2+(8-2)^2}$ = $\sqrt{1.156+81+36}$ = $\sqrt{1.273}$= 35.67
- $\sqrt{(59-3)^2+(3.5-3)^2+(20-3)^2}$ = $\sqrt{1.136+0.25+289}$ = $\sqrt{83.52}$= 289
- $\sqrt{(20-12)^2+(4.5-12)^2+(6-12)^2}$ = $\sqrt{64+90.25+36}$ = $\sqrt{190.25}$= 13.79
- $\sqrt{(34-3)^2+(11.25-3)^2+(150-3)^2}$ = $\sqrt{961+69.06+21.60}$ = $\sqrt{29.47}$= 171.68

b. Perhitungan menggunakan Python

- Library yang di butuhkan pada proses K-means clustering



- Proses memanggil data atau menginput data Cryotherapy

```
In [74]: Cryotherapy = pd.read_excel("Cryotherapy.xlsx")
```

- Melihat jumlah data yang akan digunakan dengan fungsi ".size"

```
In [51]: Cryotherapy.size
Out[51]: 630
```

- Menampilkan data dengan fungsi ".head()"

```
In [75]: Cryotherapy.head()
```

Out[75]:

	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment
0	1	35	12.00	5	1	100	0
1	1	29	7.00	5	1	96	1
2	1	50	8.00	1	3	132	0
3	1	32	11.75	7	3	750	0
4	1	67	9.25	1	1	42	0

- Menampilkan informasi data-data yang ada pada dataset Cryotherapy serta atributnya dengan fungsi ".info()"

```
In [76]: Cryotherapy.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 7 columns):
sex                   90 non-null int64
age                   90 non-null int64
Time                  90 non-null float64
Number_of_Warts       90 non-null int64
Type                  90 non-null int64
Area                  90 non-null int64
Result_of_Treatment   90 non-null int64
dtypes: float64(1), int64(6)
memory usage: 5.0 KB
```

- Melakukan pengecekan apakah terdapat deret data yang kosong menggunakan fungsi ".empty".

```
]: Cryotherapy.empty
]: False
```

Hasilnya menunjukkan false artinya didalan data set tidak terdapat deret yang kosong didalam data yang akan digunakan.

- Melakukan pemanggilan atribut untuk mengetahui jenis atribut dan menampilkan jumlah data yang missing pada setiap atribut.

```
df.dtypes
```

```
sex                    int64
age                    int64
Time                 float64
Number_of_Warts        int64
Type                   int64
Area                   int64
Result_of_Treatment    int64
dtype: object
```

```
df.isna().sum()
```

```
sex                    0
age                    0
Time                   0
Number_of_Warts        0
Type                   0
Area                   0
Result_of_Treatment    0
dtype: int64
```

- Melakukan pembersihan data menggunakan dropna()

```
da = df.dropna()
da
```

	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment
0	1	35	12.00	5	1	100	0
1	1	29	7.00	5	1	96	1
2	1	50	8.00	1	3	132	0
3	1	32	11.75	7	3	750	0
4	1	67	9.25	1	1	42	0
...	...	...	...	...	...	...	...
85	2	34	12.00	3	3	95	0
86	2	20	3.50	6	1	75	1
87	2	35	8.25	8	3	100	0
88	1	24	10.75	10	1	20	1
89	1	19	8.00	8	1	160	1

90 rows × 7 columns

- Pembersihan data menggunakan mean value

```
mean_Area = df.Area.mean()
```

```
mean_Area
```

```
85.83333333333333
```

```
df.Area.fillna(mean_Area, inplace=True)
df
```

	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment
0	1	35	12.00	5	1	100	0
1	1	29	7.00	5	1	96	1
2	1	50	8.00	1	3	132	0
3	1	32	11.75	7	3	750	0
4	1	67	9.25	1	1	42	0
...	...	...	...	...	...	...	...
85	2	34	12.00	3	3	95	0
86	2	20	3.50	6	1	75	1
87	2	35	8.25	8	3	100	0
88	1	24	10.75	10	1	20	1
89	1	19	8.00	8	1	160	1

90 rows × 7 columns

- Proses Detecting outlier

```
import statistics
from statistics import stdev
```

```
std_Area = (statistics.stdev(df.Area))
std_Area
```

```
131.73315298551697
```

```
T_Area_max = mean_Area + (2*std_Area)
T_Area_min = mean_Area - (2*std_Area)
```

```
T_Area_max
```

```
349.29963930436725
```

```
T_Area_min
```

```
-177.63297263770062
```

- Proses seleksi data

```
df.loc[:,['age','Area']]
```

	age	Area
0	35	100
1	29	96
2	50	132
3	32	750
4	67	42
...	...	...
85	34	95
86	20	75
87	35	100
88	24	20
89	19	160

- Pada seleksi data ini saya menampilkan tahun lahir dihitung dari atribut yang sudah ada sebelumnya yaitu atribut umur

```
]: from datetime import datetime
now = datetime.now()

now.year
for index, row in df.iterrows():
    df.loc[index, 'tahun_lahir']=now.year - int (row['age'])
df
```

]:

	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment	tahun_lahir
0	1	35	12.00	5	1	100	0	1987.0
1	1	29	7.00	5	1	96	1	1993.0
2	1	50	8.00	1	3	132	0	1972.0
3	1	32	11.75	7	3	750	0	1990.0
4	1	67	9.25	1	1	42	0	1955.0
...	...	...	...	...	...	...	...	...
85	2	34	12.00	3	3	95	0	1988.0
86	2	20	3.50	6	1	75	1	2002.0
87	2	35	8.25	8	3	100	0	1987.0
88	1	24	10.75	10	1	20	1	1998.0
89	1	19	8.00	8	1	160	1	2003.0

90 rows × 8 columns

- Proses transformasi menggunakan one-hot-encoding

```
]: dummies = pd.get_dummies(df.Result_of_Treatment)
dummies.columns = ['0','1']
dummies
```

]:

	0	1
0	1	0
1	0	1
2	1	0
3	1	0
4	1	0
...	...	...
85	1	0
86	0	1
87	1	0
88	0	1
89	0	1

90 rows × 2 columns

```
]: hasil = pd.concat([df,dummies],axis='columns')
hasil
```

]:

	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment	tahun_lahir	0
0	1	35	12.00	5	1	100	0	1987.0	1
1	1	29	7.00	5	1	96	1	1993.0	0
2	1	50	8.00	1	3	132	0	1972.0	1
3	1	32	11.75	7	3	750	0	1990.0	1
4	1	67	9.25	1	1	42	0	1955.0	1
...	...	...	...	...	...	...	...	...	...
85	2	34	12.00	3	3	95	0	1988.0	1
86	2	20	3.50	6	1	75	1	2002.0	0
87	2	35	8.25	8	3	100	0	1987.0	1
88	1	24	10.75	10	1	20	1	1998.0	0
89	1	19	8.00	8	1	160	1	2003.0	0

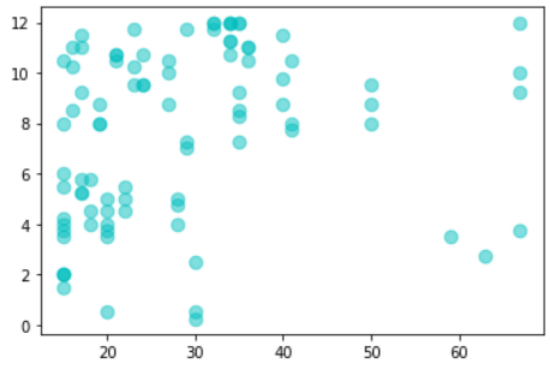
90 rows × 10 columns

- Mengimport library klasifikasi dengan K-means

```
: from sklearn.cluster import KMeans
```

- Melakukan visualisasi persebaran data

```
]: plt.scatter(Cryotherapy.age, Cryotherapy.Time, s = 75, c = "c", marker = "o", al
plt.show()
```



- Untuk melihat data random yang digunakan untuk simulasi clustering, dengan cara x_array

```
]: from sklearn.preprocessing import MinMaxScaler
```

```
]: x_array =  np.array(df_x)
print(x_array)
```

```
[[  5   1 100   0]
 [  5   1  96   1]
 [  1   3 132   0]
 [  7   3 750   0]
 [  1   1  42   0]
 [  2   2  20   1]
 [  2   1   8   0]
 [  3   3  20   0]
 [ 12   1   6   1]
 [  3   3 150   0]
 [  5   1  35   0]
 [  2   1  30   1]
 [  3   1   4   1]
 [  2   3  70   1]
 [  2   1  10   0]
 [  3   1  63   1]
 [ 12   3  72   0]
 [  2   1   6   0]
 [  1   1   6   1]
 [  1   1  80   1]
 [  2   1  70   1]
 [  1   2  60   1]
 [  3   1 100   1]
 [  1   2  80   0]
 [  2   1 115   1]
 [  3   3  95   0]
 [  2   1  75   1]
 [  5   3 100   0]
 [  3   3  20   0]
```

```
[ 6   2  80   0]
[ 8   3 115   1]
[ 1   3  95   0]
[ 11  1  75   1]
[ 6   3 100   0]
[ 8   1  20   1]
[ 9   1 160   1]
[ 2   1 100   1]
[ 5   1  96   0]
[ 4   3 132   0]
[ 12  3 750   0]
[ 7   1  42   0]
[ 5   2  20   1]
[ 4   1   8   0]
[ 11  3  20   0]
[ 3   1   6   1]
[ 1   3 150   0]
[ 7   1  35   0]
[ 11  1  30   1]
[ 11  1   4   1]
[ 10  3  70   1]
[ 12  1  10   0]
[ 10  1  63   1]
[ 7   3  72   0]
[ 7   1   6   0]
[ 5   1   6   1]
[ 1   1  80   1]
[ 2   1  70   1]
[ 3   2  60   1]
[ 9   1 100   1]
[ 9   2  80   0]
[ 10  1 115   1]
[ 3   3  95   0]
[ 6   1  75   1]
[ 8   3 100   0]
[ 10  1  20   1]
[ 8   1 160   1]]
```

```
]: scaler = MinMaxScaler()
   x_scaled = scaler.fit_transform(x_array)
   x_scaled
```

```
]: array([[0.36363636, 0.        , 0.12868633, 0.        ],
          [0.36363636, 0.        , 0.1233244 , 1.        ],
          [0.        , 1.        , 0.17158177, 0.        ],
          [0.54545455, 1.        , 1.        , 0.        ],
          [0.        , 0.        , 0.05093834, 0.        ],
          [0.09090909, 0.5       , 0.02144772, 1.        ],
          [0.09090909, 0.        , 0.00536193, 0.        ],
          [0.18181818, 1.        , 0.02144772, 0.        ],
          [1.        , 0.        , 0.00268097, 1.        ],
          [0.18181818, 1.        , 0.19571046, 0.        ],
          [0.36363636, 0.        , 0.04155496, 0.        ],
          [0.09090909, 0.        , 0.03485255, 1.        ],
          [0.18181818, 0.        , 0.        , 1.        ],
          [0.09090909, 1.        , 0.08847185, 1.        ],
          [0.09090909, 0.        , 0.0080429 , 0.        ],
          [0.18181818, 0.        , 0.07908847, 1.        ],
          [1.        , 1.        , 0.09115282, 0.        ],
          [0.09090909, 0.        , 0.00268097, 0.        ],
          [0.        , 0.        , 0.00268097, 1.        ],
```

- Kemudian melakukan fungsi kmeans_fit() dengan menambahkan parameter jumlah cluster (n_cluster=15)

```
: kmeans = KMeans(n_clusters = 15, random_state=0)
  kmeans.fit(x_scaled)
```

```
: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
         n_clusters=15, n_init=10, n_jobs=None, precompute_distances='auto',
         random_state=0, tol=0.0001, verbose=0)
```

```
: print(kmeans.cluster_centers_)
```

```
[[ 8.98989899e-01 -5.55111512e-17  6.06196008e-02  1.00000000e+00]
 [ 1.23966942e-01  1.00000000e+00  1.23689983e-01  0.00000000e+00]
 [ 1.41414141e-01  1.11022302e-16  6.71730712e-02  1.00000000e+00]
 [ 4.18181818e-01 -5.55111512e-17  4.43699732e-02  0.00000000e+00]
 [ 1.51515152e-01  5.00000000e-01  4.82573727e-02  1.00000000e+00]
 [ 9.54545455e-01  1.00000000e+00  8.37801609e-02  0.00000000e+00]
 [ 4.09090909e-01  1.00000000e+00  1.00000000e+00  0.00000000e+00]
 [ 6.11570248e-01 -5.55111512e-17  1.19790397e-01  1.00000000e+00]
 [ 8.18181818e-01  1.00000000e+00  1.08579088e-01  1.00000000e+00]
 [ 1.00000000e+00  0.00000000e+00  8.04289544e-03  0.00000000e+00]
 [ 6.06060606e-02  8.33333333e-02  2.94906166e-02  0.00000000e+00]
 [ 5.90909091e-01  5.00000000e-01  1.01876676e-01  0.00000000e+00]
 [ 9.09090909e-02  1.00000000e+00  8.84718499e-02  1.00000000e+00]
 [ 4.72727273e-01  1.00000000e+00  1.13672922e-01  0.00000000e+00]
 [ 1.00000000e+00  1.00000000e+00  1.00000000e+00  0.00000000e+00]]
```

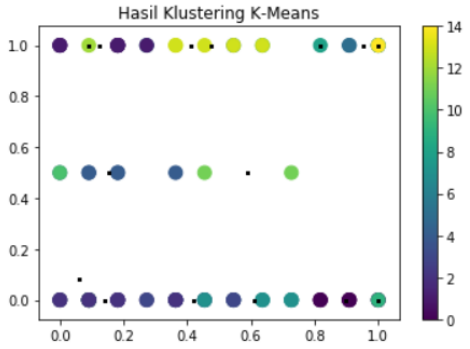
- Untuk melihat kmeans label

```
print(kmeans.labels_)
df["kluster"] = kmeans.labels_
```

```
[ 3  2  1  6 10  4 10  1  0  1  3  2  2 12 10  2  5 10  2  2  2  4  2 10
  2  1  2 13  1  7  7  7  5  6  5  4  3  1  2  1  3  0  2  8 10  2 13  3
  7  7  7  4  0 11  8  1  0 13  7  7  2  3  1 14  3  4  3  5  2  1  3  0
  0  8  9  0 13  3  2  2  2  4  7 11  0  1  7 13  0  7]
```

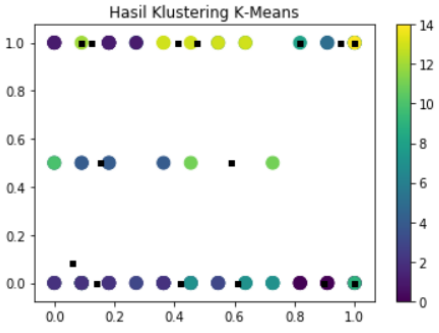
- Melakukan Hasil clustering yang di visualisasikan (cluster 5)

```
output = plt.scatter(x_scaled[:,0], x_scaled[:,1], s = 100, c = df.kluster, mark
centers = kmeans.cluster_centers_
plt.scatter(centers[:,0], centers[:,1], c='black', s=5, alpha=1 , marker="s");
plt.title("Hasil Klustering K-Means")
plt.colorbar (output)
plt.show()
```



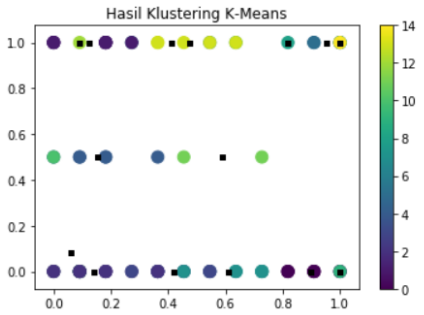
- Melakukan Hasil clustering yang di visualisasikan (cluster 10)

```
output = plt.scatter(x_scaled[:,0], x_scaled[:,1], s = 100, c = df.kluster, mark
centers = kmeans.cluster_centers_
plt.scatter(centers[:,0], centers[:,1], c='black', s=10, alpha=1 , marker="s");
plt.title("Hasil Klustering K-Means")
plt.colorbar (output)
plt.show()
```



- Melakukan Hasil clustering yang di visualisasikan (cluster 15)

```
output = plt.scatter(x_scaled[:,0], x_scaled[:,1], s = 100, c = df.kluster, mark
centers = kmeans.cluster_centers_
plt.scatter(centers[:,0], centers[:,1], c='black', s=15, alpha=1 , marker="s");
plt.title("Hasil Klustering K-Means")
plt.colorbar (output)
plt.show()
```



- Melakukan pengecekan apakah dataset terdapat deret data dengan fungsi ".empty"

```
Cryotherapy.empty
```

```
False
```

Hasil outputnya false artinya tidak terdapat deret data yang kosong di dalam data set Cryotherapy

- Kemudian Menentukan variable independent dan dependen dari data set dengan cara dibawah ini

```
Cryotherapy.empty
```

```
False
```

```
x = Cryotherapy.drop(["Result_of_Treatment"], axis = 1)
x.head()
```

	sex	age	Time	Number_of_Warts	Type	Area
0	1	35	12.00	5	1	100
1	1	29	7.00	5	1	96
2	1	50	8.00	1	3	132
3	1	32	11.75	7	3	750
4	1	67	9.25	1	1	42

```
y = Cryotherapy["Result_of_Treatment"]
y.head()
```

```
0    0
1    1
2    0
3    0
4    0
Name: Result_of_Treatment, dtype: int64
```

- Setelah itu melakukan proses clustering Naïve Bayes menggunakan fungsi " Train Test Split" fungsi ini berguna untuk membagi dataset menjadi data training dan data testing

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, rando
```

Data training 80% , sedangkan data testing 20 %. Yang artinya dari 630 data tadi yang sudah dicek,504 data training dan 120 data testing.

Proses Penghitungan Akurasi

- Melakukan prediksi pada data training dan data testing,menggunakan library GaussianNB(), kemudian memsakukan data training dan memanggilnya

```
]: from sklearn.naive_bayes import GaussianNB
   modelnb = GaussianNB()
   nbtrain = modelnb.fit(x_train, y_train)

   nbtrain.class_count_
```

```
]: array([33., 39.])
```

- Menentukan hasil prediksi dari x_test

```
]: y_pred = nbtrain.predict(x_test)
   y_pred
```

```
]: array([1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1], dtype=int64)
```

- Melakukan confussion matrix dengan menggunakan y_pred, y_test

```
: from sklearn.metrics import confusion_matrix

  confusion_matrix(y_test, y_pred)
```

```
: array([[7, 2],
         [1, 8]], dtype=int64)
```

- Setelah mendapatkan hasil confussion matrix selanjutnya melakukan merapikan hasil dari confusion matrix

```
: import pandas as pd
  y_actual1 = pd.Series([1,0,1,0,1,0,1,0,1,0,0,1,1,0,1,1,0,0],name ="actual")

  y_pred1 = pd.Series([1,1,1,0,1,1,0,1,0,0,1,1,0,1,0,1,0,0,1],name = "prediction")

  df_confusion = pd.crosstab(y_actual1, y_pred1)

  df_confusion
```

:

prediction	0	1
actual		
0	4	5
1	4	5

- Tahap terakhir yaitu melakukan perhitungan nilai akurasi dari data set Cryotherapy dengan menggunakan nilai akurasi dari klarifikasi naïve bayes

```
: from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.78      0.82         9
           1       0.80      0.89      0.84         9

    accuracy                           0.83        18
   macro avg       0.84      0.83      0.83        18
weighted avg       0.84      0.83      0.83        18
```

6. Analisa Hasil

Setelah melakukan percobaan menggunakan *python* (*Jupyter Notebok*) dari hasil analisis menggunakan metode *naïve bayes* pada data *Cryotherapy* menunjukkan bahwa ada beberapa prediksi bisa dilihat dari gambar di bawah ini :

```
In [18]: import pandas as pd
y_actual1 = pd.Series([1,0,1,0,1,0,1,0,1,0,0,1,1,0,1,1,0,0],name ="actual")

y_pred1 = pd.Series([1,1,1,0,1,1,0,1,0,0,1,1,0,1,0,1,0,0,1],name = "prediction")

df_confusion = pd.crosstab(y_actual1, y_pred1)

df_confusion
```

Out[18]:

prediction	0	1
actual		
0	4	5
1	4	5

Dari gambar diatas penjabarannya adalah prediksi treatment atau perawatan dinyatakan gagal berjumlah 4, ,prediksi treatment gagal/actual berhasil 1,dan prediksi *treatment* berhasil sebanyak 5 , karena Sebagian besar prediksi dan hasilnya sesuai maka dapat dikatan bahwa perawatan kulit menggunakan *Cryotherapy* baik digunakan.

Kelebihan dari dataset *Cryotherapy* adalah jelas dalam datanya, dan tidak ada ada terdapat deret data yang kosong dalam dataset, sehingga memudahkan dalam proses klasifikasi dan prediksi menggunakan *K-means clustering*, dalam dataset tersebut sudah jelas datanya hanya saja ada satu kekurangnya dalam implementasi di *python* jumlah dari prediksi gagal kurang hampir sebanding dengan dengan prediksi berhasil hanya selisih satu.

Hasil akurasi dari percobaan menggunakan *python* sebagai berikut

```
96]: from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.78      0.82         9
           1       0.80      0.89      0.84         9

    accuracy                           0.83        18
   macro avg       0.84      0.83      0.83        18
weighted avg       0.84      0.83      0.83        18
```