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

df = pd.read_csv("lv2-2305-2.csv")
df.head(3)
```

Out[1]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT
0	1	337	118	B	4.5	4.5	9.65	1	Admitted
1	2	324	107	B	4.0	4.5	8.87	1	Admitted
2	3	316	104	C	3.0	3.5	8.00	1	Denied

In [2]:
```
df.shape
```

Out[2]:
```
(400, 9)
```

In [3]:
```
admit = df.copy()
```

In [4]:
```
admit_p = admit.loc[admit["ADMIT"]=="Admitted"]
admit_f = admit.loc[admit["ADMIT"]=="Denied"]
admit_p.shape, admit_f.shape
```

Out[4]:
```
((180, 9), (220, 9))
```

In [7]:
```
admit_p["GRE_R"] = admit_p["GRE"].rank(method = 'max')
admit_p["TOEFL_R"] = admit_p["TOEFL"].rank(method = 'max')
admit_p["CGPA_R"] = admit_p["CGPA"].rank(method = 'max')

admit_p.head(3)
```

```
C:₩Users₩Hyemin₩AppData₩Local₩Temp₩ipykernel_18540₩1196072363.py:1: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  admit_p["GRE_R"] = admit_p["GRE"].rank(method = 'max')
C:₩Users₩Hyemin₩AppData₩Local₩Temp₩ipykernel_18540₩1196072363.py:2: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  admit_p["TOEFL_R"] = admit_p["TOEFL"].rank(method = 'max')
C:₩Users₩Hyemin₩AppData₩Local₩Temp₩ipykernel_18540₩1196072363.py:3: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  admit_p["CGPA_R"] = admit_p["CGPA"].rank(method = 'max')
```

Out[7]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT	GRE_R	TOEFL_R	CGPA_R
0	1	337	118	B	4.5	4.5	9.65	1	Admitted	165.0	163.0	163.0
1	2	324	107	B	4.0	4.5	8.87	1	Admitted	84.0	28.0	55.0
3	4	322	110	C	3.5	2.5	8.67	1	Admitted	61.0	72.0	27.0

In [26]:
```
admit_f["GRE_R"] = admit_f["GRE"].rank(method = 'max')
admit_f["TOEFL_R"] = admit_f["TOEFL"].rank(method = 'max')
admit_f["CGPA_R"] = admit_f["CGPA"].rank(method = 'max')

admit_f.head(3)
```

```
C:₩Users₩Hyemin₩AppData₩Local₩Temp₩ipykernel_18540₩3902571758.py:1: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  admit_f["GRE_R"] = admit_f["GRE"].rank(method = 'max')
C:₩Users₩Hyemin₩AppData₩Local₩Temp₩ipykernel_18540₩3902571758.py:2: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  admit_f["TOEFL_R"] = admit_f["TOEFL"].rank(method = 'max')
C:₩Users₩Hyemin₩AppData₩Local₩Temp₩ipykernel_18540₩3902571758.py:3: 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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  admit_f["CGPA_R"] = admit_f["CGPA"].rank(method = 'max')
```

Out[26]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT	GRE_R	TOEFL_R	CGPA_R
2	3	316	104	C	3.0	3.5	8.00	1	Denied	171.0	119.0	66.0
4	5	314	103	D	2.0	3.0	8.21	0	Denied	150.0	98.0	107.0
7	8	308	101	D	3.0	4.0	7.90	0	Denied	96.0	71.0	53.0

In [14]:
```
admit_p.iloc[:,9:12].corr('spearman')
```

Out[14]:

	GRE_R	TOEFL_R	CGPA_R
GRE_R	1.000000	0.784158	0.777091
TOEFL_R	0.784158	1.000000	0.738595
CGPA_R	0.777091	0.738595	1.000000

In [15]:
```
admit_p.loc[:,"GRE_R":].corr('spearman')
```

Out[15]:

	GRE_R	TOEFL_R	CGPA_R
GRE_R	1.000000	0.784158	0.777091
TOEFL_R	0.784158	1.000000	0.738595
CGPA_R	0.777091	0.738595	1.000000

In [31]:
```
A = admit_p.loc[:,"GRE_R":].corr('spearman').iloc[2,:2].max()
A_VAR = admit_p.loc[:,"GRE_R":].corr('spearman').iloc[2,2].idxmax()
A, A_VAR
```

Out[31]:
```
(0.7770913247704572, 'GRE_R')
```

In [37]:
```
B = admit_f.loc[:,"GRE_R":].corr('spearman').iloc[2,:2].max()
B_VAR = admit_f.loc[:,"GRE_R":].corr('spearman').iloc[2,:2].idxmax()
B, B_VAR
```

Out[37]:
```
(0.616724306652575, 'TOEFL_R')
```

In [38]:
```
admit_f.loc[:,"GRE_R":].corr('spearman')
```

Out[38]:

	GRE_R	TOEFL_R	CGPA_R
GRE_R	1.000000	0.613201	0.595403
TOEFL_R	0.613201	1.000000	0.616724
CGPA_R	0.595403	0.616724	1.000000

In [42]:
```
round(abs(B - A),2)
```

Out[42]:
```
0.16
```

In [43]:
```
########## 2번
```

In [44]:
```
from sklearn.linear_model import LogisticRegression
```

In [94]:
```
df2 = admit.copy()
df2.head(3
        )
```

Out[94]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT
0	1	337	118	B	4.5	4.5	9.65	1	Admitted
1	2	324	107	B	4.0	4.5	8.87	1	Admitted
2	3	316	104	C	3.0	3.5	8.00	1	Denied

In [95]:
```
q1 = df2["TOEFL"].quantile(0.25)
q3 = df2["TOEFL"].quantile(0.75)
q1, q3
```

Out[95]:
```
(103.0, 112.0)
```

In [96]:
```
df2[(df2["TOEFL"]>q1) &(df2["TOEFL"]<q3)]
```

Out[96]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT
1	2	324	107	B	4.0	4.5	8.87	1	Admitted
2	3	316	104	C	3.0	3.5	8.00	1	Denied
3	4	322	110	C	3.5	2.5	8.67	1	Admitted
6	7	321	109	C	3.0	4.0	8.20	1	Admitted
9	10	323	108	C	3.5	3.0	8.60	0	Denied
...	...	...	...	...	...	...	...	...	...
389	390	320	108	C	3.5	4.0	8.44	1	Admitted
391	392	318	106	C	2.0	3.0	8.65	0	Denied
393	394	317	104	D	3.0	3.0	8.76	0	Admitted
394	395	329	111	B	4.5	4.0	9.23	1	Admitted
395	396	324	110	C	3.5	3.5	9.04	1	Admitted

189 rows × 9 columns

In [97]:
```
df2.loc[(df2["TOEFL"]>q1) &(df2["TOEFL"]<q3)]
```

Out[97]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT
1	2	324	107	B	4.0	4.5	8.87	1	Admitted
2	3	316	104	C	3.0	3.5	8.00	1	Denied
3	4	322	110	C	3.5	2.5	8.67	1	Admitted
6	7	321	109	C	3.0	4.0	8.20	1	Admitted
9	10	323	108	C	3.5	3.0	8.60	0	Denied
...	...	...	...	...	...	...	...	...	...
389	390	320	108	C	3.5	4.0	8.44	1	Admitted
391	392	318	106	C	2.0	3.0	8.65	0	Denied
393	394	317	104	D	3.0	3.0	8.76	0	Admitted
394	395	329	111	B	4.5	4.0	9.23	1	Admitted
395	396	324	110	C	3.5	3.5	9.04	1	Admitted

189 rows × 9 columns

In [98]:
```
df2 = df2.loc[(df2["TOEFL"]>q1) &(df2["TOEFL"]<q3)]
df2 = pd.DataFrame(df2)
df2.head(3)
```

Out[98]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT
1	2	324	107	B	4.0	4.5	8.87	1	Admitted
2	3	316	104	C	3.0	3.5	8.00	1	Denied
3	4	322	110	C	3.5	2.5	8.67	1	Admitted

In [99]:
```
df2["ADMIT_S"] = np.where(df2["ADMIT"]=="Admitted",1,0)
df2.head()
```

Out[99]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT	ADMIT_S
1	2	324	107	B	4.0	4.5	8.87	1	Admitted	1
2	3	316	104	C	3.0	3.5	8.00	1	Denied	0
3	4	322	110	C	3.5	2.5	8.67	1	Admitted	1
6	7	321	109	C	3.0	4.0	8.20	1	Admitted	1
9	10	323	108	C	3.5	3.0	8.60	0	Denied	0

In [100]:
```
df2["ADMIT_S"].dtype
```

Out[100]:
```
dtype('int32')
```

In [101]:
```
model = LogisticRegression(solver="newton-cg", C=100000, random_state = 1234).fit(X = df2[["GRE","TOEFL","SOP","LOR","CGPA"]],
                                                                                  y = df2["ADMIT_S"])
model
```

Out[101]:
```
▾                    LogisticRegression
LogisticRegression(C=100000, random_state=1234, solver='newton-cg')
```

In [102]:
```
df2["pred"] = model.predict(df2[["GRE","TOEFL","SOP","LOR","CGPA"]])
df2.head(3)
```

Out[102]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT	ADMIT_S	pred
1	2	324	107	B	4.0	4.5	8.87	1	Admitted	1	1
2	3	316	104	C	3.0	3.5	8.00	1	Denied	0	0
3	4	322	110	C	3.5	2.5	8.67	1	Admitted	1	1

In [103]:
```
(df2[df2["ADMIT_S"]==df2["pred"]].shape[0]) / df2.shape[0]
```

Out[103]:
```
0.8148148148148148
```

In [104]:
```
from sklearn.metrics import accuracy_score
```

In [107]:
```
(accuracy_score(df2["ADMIT_S"], df2["pred"])*100).round(2)
```

Out[107]:
```
81.48
```

In [110]:
```
############ 3번

df3 = admit.copy()
df3.head(3)
```

Out[110]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT
0	1	337	118	B	4.5	4.5	9.65	1	Admitted
1	2	324	107	B	4.0	4.5	8.87	1	Admitted
2	3	316	104	C	3.0	3.5	8.00	1	Denied

In [115]:
```
df3 = df3[df3["ADMIT"]=="Denied"].reset_index(drop = True)
df3.head(3)
```

Out[115]:

	REG_NO	GRE	TOEFL	UNIV_RATING	SOP	LOR	CGPA	RESEARCH	ADMIT
0	3	316	104	C	3.0	3.5	8.00	1	Denied
1	5	314	103	D	2.0	3.0	8.21	0	Denied
2	8	308	101	D	3.0	4.0	7.90	0	Denied

In [117]:
```
df3_dum = pd.get_dummies(df3,drop_first = True)
df3_dum.head(3)
```

Out[117]:

	REG_NO	GRE	TOEFL	SOP	LOR	CGPA	RESEARCH	UNIV_RATING_B	UNIV_RATING_C	UNIV_RATING_D	UNIV_RATING_E
0	3	316	104	3.0	3.5	8.00	1	False	True	False	False
1	5	314	103	2.0	3.0	8.21	0	False	False	True	False
2	8	308	101	3.0	4.0	7.90	0	False	False	True	False

In [118]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [130]:
```
df3_dum.iloc[:,1:6]
```

	GRE	TOEFL	SOP	LOR	CGPA
0	316	104	3.0	3.5	8.00
1	314	103	2.0	3.0	8.21
2	308	101	3.0	4.0	7.90
3	302	102	2.0	1.5	8.00
4	323	108	3.5	3.0	8.60
...	...	...	...	...	...
215	307	105	2.0	3.5	8.10
216	296	97	1.5	2.0	7.80
217	314	102	2.0	2.5	8.24
218	318	106	2.0	3.0	8.65
219	312	103	3.5	4.0	8.78

220 rows × 5 columns

In [131]:
```
MinMaxScaler().fit_transform(df3_dum.iloc[:,1:6])
```

Out[131]:
```
array([[0.66666667, 0.82539683, 0.5       , 0.625     , 0.49586777],
       [0.61538462, 0.80952381, 0.25      , 0.5       , 0.58264463],
       [0.46153846, 0.77777778, 0.5       , 0.75      , 0.45454545],
       ...,
       [0.61538462, 0.79365079, 0.25      , 0.375     , 0.59504132],
       [0.71794872, 0.85714286, 0.25      , 0.5       , 0.76446281],
       [0.56410256, 0.80952381, 0.625     , 0.75      , 0.81818182]])
```

In [134]:
```
var3 = ["GRE", "TOEFL", "SOP", "LOR", "CGPA"]
df3_s = pd.DataFrame(MinMaxScaler().fit_transform(df3_dum[var3]), columns = var3)
df3_s.head(3)
```

Out[134]:

	GRE	TOEFL	SOP	LOR	CGPA
0	0.666667	0.825397	0.50	0.625	0.495868
1	0.615385	0.809524	0.25	0.500	0.582645
2	0.461538	0.777778	0.50	0.750	0.454545

In [135]:
```
df3_dum.head(3)
```

Out[135]:

	REG_NO	GRE	TOEFL	SOP	LOR	CGPA	RESEARCH	UNIV_RATING_B	UNIV_RATING_C	UNIV_RATING_D	UNIV_RATING_E
0	3	316	104	3.0	3.5	8.00	1	False	True	False	False
1	5	314	103	2.0	3.0	8.21	0	False	False	True	False
2	8	308	101	3.0	4.0	7.90	0	False	False	True	False

In [144]:
```
df3_F = pd.concat([df3_s,df3_dum.iloc[:,6:]],axis = 1)
df3_F.head(3)
```

Out[144]:

	GRE	TOEFL	SOP	LOR	CGPA	RESEARCH	UNIV_RATING_B	UNIV_RATING_C	UNIV_RATING_D	UNIV_RATING_E
0	0.666667	0.825397	0.50	0.625	0.495868	1	False	True	False	False
1	0.615385	0.809524	0.25	0.500	0.582645	0	False	False	True	False
2	0.461538	0.777778	0.50	0.750	0.454545	0	False	False	True	False

In [142]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

In [165]:
```
label4 = KMeans(n_clusters=4, random_state = 1234, n_init = 50, max_iter = 300).fit_predict(X=df3_F)
label4
```

```
C:₩Users₩Hyemin₩anaconda3₩Lib₩site-packages₩sklearn₩cluster₩_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_N
UM_THREADS=1.
  warnings.warn(
```

Out[165]:
```
array([0, 1, 1, 3, 2, 0, 0, 0, 2, 2, 0, 2, 2, 0, 3, 1, 3, 1, 1, 0, 1, 3,
       3, 1, 0, 1, 1, 1, 0, 2, 2, 2, 1, 3, 1, 1, 2, 1, 1, 2, 2, 2, 1, 0,
       2, 1, 0, 1, 1, 3, 0, 2, 1, 2, 0, 1, 2, 1, 1, 2, 3, 1, 0, 1, 1, 1,
       0, 0, 2, 2, 0, 0, 1, 0, 0, 2, 3, 1, 0, 3, 2, 3, 0, 2, 1, 2, 1, 1,
       3, 3, 3, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 2, 2, 0, 1,
       2, 1, 1, 0, 1, 0, 2, 1, 1, 1, 1, 1, 2, 2, 0, 1, 1, 2, 3, 3, 1, 0,
       1, 2, 1, 0, 1, 1, 2, 1, 0, 2, 1, 0, 1, 1, 3, 3, 3, 1, 1, 1, 0, 1,
       1, 1, 3, 1, 1, 2, 1, 1, 1, 0, 1, 0, 2, 3, 2, 1, 1, 1, 3, 0, 1, 1,
       2, 2, 1, 1, 1, 0, 0, 2, 2, 1, 1, 3, 1, 3, 3, 2, 0, 0, 1, 2, 1, 1,
       1, 1, 1, 0, 3, 3, 3, 1, 1, 1, 1, 3, 3, 3, 0, 2, 1, 1, 1, 1, 2, 2])
```

In [166]:
```
silhouette_score(df3_F, labels = label4)
```

Out[166]:
```
0.48982146500189O7
```

In [163]:
```
sil = []

for i in range(4,7):
    label = KMeans(n_clusters=i, random_state = 1234, n_init = 50, max_iter = 300).fit_predict(X=df3_F)
    sil = sil + [silhouette_score(df3_F, labels = label)]
sil
```

```
C:₩Users₩Hyemin₩anaconda3₩Lib₩site-packages₩sklearn₩cluster₩_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_N
UM_THREADS=1.
  warnings.warn(
C:₩Users₩Hyemin₩anaconda3₩Lib₩site-packages₩sklearn₩cluster₩_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_N
UM_THREADS=1.
  warnings.warn(
C:₩Users₩Hyemin₩anaconda3₩Lib₩site-packages₩sklearn₩cluster₩_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_N
UM_THREADS=1.
  warnings.warn(
```

Out[163]:
```
[0.4898214650018907, 0.5090779061903452, 0.5556812882602017]
```

In [167]:
```
label5 = KMeans(n_clusters=5, random_state = 1234, n_init = 50, max_iter = 300).fit_predict(X=df3_F)
label5
```

```
C:₩Users₩Hyemin₩anaconda3₩Lib₩site-packages₩sklearn₩cluster₩_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_N
UM_THREADS=1.
  warnings.warn(
```

Out[167]:
```
array([0, 3, 3, 1, 4, 0, 0, 0, 4, 4, 0, 4, 4, 0, 1, 2, 1, 3, 2, 0, 3, 1,
       1, 3, 0, 2, 2, 2, 0, 4, 4, 4, 3, 1, 3, 3, 4, 2, 2, 4, 1, 4, 2, 0,
       4, 2, 0, 3, 2, 1, 0, 4, 3, 4, 0, 2, 4, 3, 2, 4, 1, 3, 0, 3, 3, 3,
       0, 0, 4, 1, 0, 0, 3, 0, 0, 4, 1, 3, 0, 1, 4, 1, 0, 1, 2, 4, 3, 3,
       1, 1, 1, 4, 4, 4, 4, 2, 3, 2, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 0, 2,
       4, 3, 3, 0, 3, 0, 4, 3, 3, 3, 3, 3, 4, 4, 0, 3, 3, 4, 1, 1, 3, 0,
       3, 4, 3, 0, 3, 2, 4, 2, 0, 4, 3, 0, 2, 3, 1, 1, 1, 3, 3, 3, 0, 3,
       3, 3, 1, 2, 3, 4, 3, 3, 2, 0, 3, 0, 4, 1, 4, 3, 3, 3, 1, 0, 3, 3,
       4, 4, 3, 3, 2, 0, 0, 4, 4, 3, 3, 1, 3, 1, 1, 4, 0, 0, 2, 4, 3, 3,
       2, 3, 3, 0, 1, 1, 1, 3, 3, 3, 3, 1, 1, 1, 0, 4, 3, 3, 3, 3, 4, 4])
```

In [168]:
```
label6 = KMeans(n_clusters=6, random_state = 1234, n_init = 50, max_iter = 300).fit_predict(X=df3_F)
label6
```

```
C:₩Users₩Hyemin₩anaconda3₩Lib₩site-packages₩sklearn₩cluster₩_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_N
UM_THREADS=1.
  warnings.warn(
```

Out[168]:
```
array([2, 4, 4, 3, 1, 2, 2, 2, 1, 1, 2, 1, 1, 2, 5, 0, 3, 4, 0, 2, 4, 3,
       3, 4, 2, 0, 0, 0, 5, 1, 1, 1, 4, 3, 4, 4, 1, 0, 0, 1, 5, 1, 0, 2,
       1, 0, 2, 4, 0, 3, 2, 1, 4, 1, 5, 0, 1, 4, 0, 1, 5, 4, 2, 4, 4, 4,
       2, 2, 1, 1, 5, 2, 4, 2, 5, 1, 5, 4, 2, 5, 1, 5, 2, 1, 0, 1, 4, 4,
       3, 3, 3, 1, 1, 1, 1, 0, 4, 0, 1, 1, 1, 4, 4, 4, 4, 4, 1, 1, 2, 0,
       1, 4, 4, 2, 4, 2, 1, 4, 4, 4, 4, 4, 1, 1, 2, 4, 4, 1, 3, 3, 4, 2,
       4, 1, 4, 2, 4, 0, 1, 0, 2, 1, 4, 2, 0, 4, 3, 3, 3, 4, 4, 4, 2, 4,
       4, 4, 3, 0, 4, 1, 4, 4, 0, 2, 4, 2, 1, 5, 1, 4, 4, 4, 3, 2, 4, 4,
       1, 1, 4, 4, 0, 2, 5, 1, 1, 4, 4, 3, 4, 3, 3, 1, 2, 5, 0, 1, 4, 4,
       0, 4, 4, 2, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 2, 1, 4, 4, 4, 4, 1, 1])
```

In [173]:
```
(pd.DataFrame(label5).value_counts().max()) / (pd.DataFrame(label5).value_counts().min())
```

Out[173]:
```
3.0416666666666665
```

In [188]:
```
(pd.DataFrame(label6).value_counts().max()) / (pd.DataFrame(label6).value_counts().min())
```

Out[188]:
```
5.615384615384615
```

In [191]:
```
res = 5 + silhouette_score(df3_F, labels = label5)
```

In [192]:
```
round(res,2)
```

Out[192]:
```
5.509
```

In [ ]: