Notebook

Load Package

[ ] ↳ *2 cells hidden*

Load Data and Explore

```
df = pd.read_csv("nw_korea_1.csv")
#df = pd.read_csv("Data/networkanalysis_cum.csv")
```

```
df.rename(columns = {'author':'Usernames'}, inplace = True)
```

```
view=df.groupby(['Celebrity','Usernames']).size().reset_index(name='Freq')
#view
```

```
a=["Celebrity","Usernames"]
data = view[a]
#data = df[a]
data.shape
```

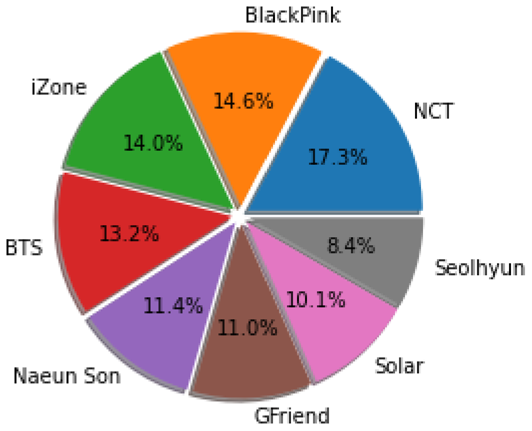
```
    (2021, 2)
```

```
piecount = data.Celebrity.value_counts()
piecount = pd.DataFrame(piecount)
```

```
labels=piecount.index
explode = []
for k in piecount.index:
        explode.append(0.05)
```

```
pie = plt.pie(piecount.values, labels=labels, explode=explode, shadow=True, autopct='%1.1f%%')
```

```
    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: MatplotlibDeprecationWarning: Non-1D inputs to pie() are currently
      """
```



```
print(*data.Celebrity.unique(),sep="\n")
```

```
    BTS
    BlackPink
    GFriend
    NCT
    Naeun Son
    Seolhyun
    Solar
    iZone
```

```
data.shape
```

```
    (2021, 2)
```

```
print("Number of Celebrities: %0.0f" %len(data.Celebrity.unique()))
print("Number of Users: %0.0f" %len(data.Usernames.unique()))
```

```
    Number of Celebrities: 8
    Number of Users: 1520
```

```
print("The percentage of unique values: {:.2%}".format(len(data.Usernames.unique())/len(data.Usernames)))
```

```
    The percentage of unique values: 75.21%
```

Generate Adjacency Matrix

```
df_merge = data.merge(data, on='Usernames')
results = pd.crosstab(df_merge.Celebrity_x, df_merge.Celebrity_y)
np.fill_diagonal(results.values, 0)
network_table=results
network_table
```

Celebrity_y	BTS	BlackPink	GFriend	NCT	Naeun Son	Seolhyun	Solar	iZone
Celebrity_x								
BTS	0	22	32	35	25	14	34	47
BlackPink	22	0	36	35	35	25	21	29
GFriend	32	36	0	43	49	36	41	60
NCT	35	35	43	0	40	31	50	55
Naeun Son	25	35	49	40	0	41	41	57
Seolhyun	14	25	36	31	41	0	32	39
Solar	34	21	41	50	41	32	0	55
iZone	47	29	60	55	57	39	55	0

Fit NetworkX

```
#graph=nx.from_numpy_matrix(np.matrix)
```

```
#graph=nx.from_numpy_matrix(np_matrix)
graph=nx.from_pandas_adjacency(network_table)
print(nx.info(graph))
```

```
Name: 
Type: Graph
Number of nodes: 8
Number of edges: 28
Average degree:   7.0000
```

```
edges,weights  =  zip(*nx.get_edge_attributes(graph,'weight').items())
pos=nx.spring_layout(graph,scale=2)
nx.draw(graph,
                                   pos,
                                   with_labels=True,
                                   node_size=600,
                                   node_color="mistyrose",
                                   edgelist=edges,
                                   edge_color=weights,
                                   edge_cmap=plt.cm.GnBu,
                                   style="solid",
                                   width=2.5)
```

```
setup = Node2Vec(graph,dimensions=100, walk_length=50, num_walks=4)
model = setup.fit(window=4, min_count=1)
```

```
Computing transition probabilities: 100%          8/8 [00:02<00:00, 3.13it/s]
Generating walks (CPU: 1): 100%|██████████| 4/4 [00:00<00:00, 42.43it/s]
```

```
#vocab, vectors = model.wv.key_to_index, model.wv.get_normed_vectors()
vocab, vectors = model.wv.vocab, model.wv.vectors

# get node name and embedding vector index.
name_index = np.array([(v[0], v[1].index) for v in vocab.items()]) #.index

# init dataframe using embedding vectors and set index as node name
node2vec_output = pd.DataFrame(vectors[name_index[:,1].astype(int)])
node2vec_output.index = name_index[:,0]
```

```
#node2vec_output
```

```
node2vec_output.shape
```

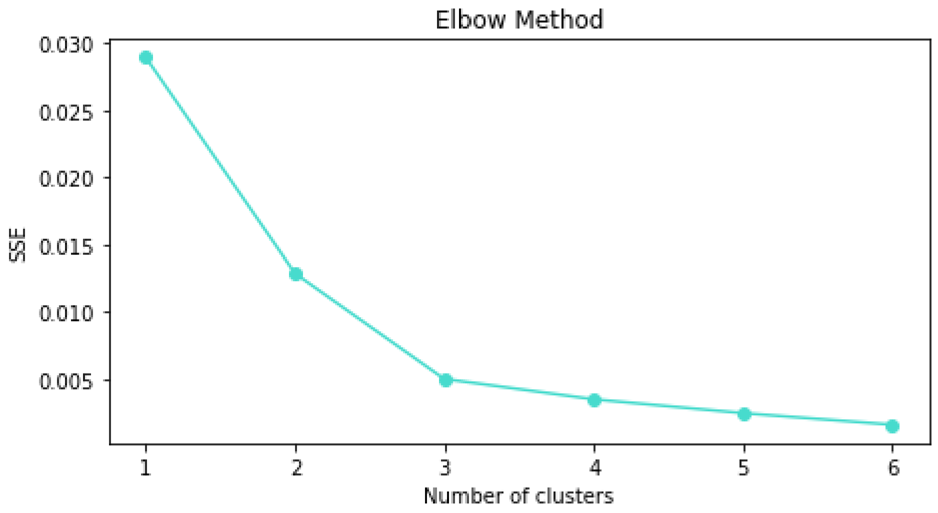
```
(8, 100)
```

```
#node2vec_output.to_csv("node2vec_k3.csv")
```

K-means: find "K"

```
SSE = []
for i in range(1,(len(node2vec_output.index)-1)):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=50, random_state=42)
    kmeans.fit(node2vec_output)
```

```
        kmeans.fit(node2vec_output)
        SSE.append(kmeans.inertia_)
plt.plot(range(1,(len(node2vec_output.index)-1)),  SSE,"o-",color="#47DBCD")
plt.title('Elbow  Method')
plt.xlabel('Number  of  clusters')
plt.ylabel('SSE')
plt.subplots_adjust(left=0.25,  bottom=0.8,  right=1.2,  top=1.5)
plt.show()
```



K-means: training and subsampling

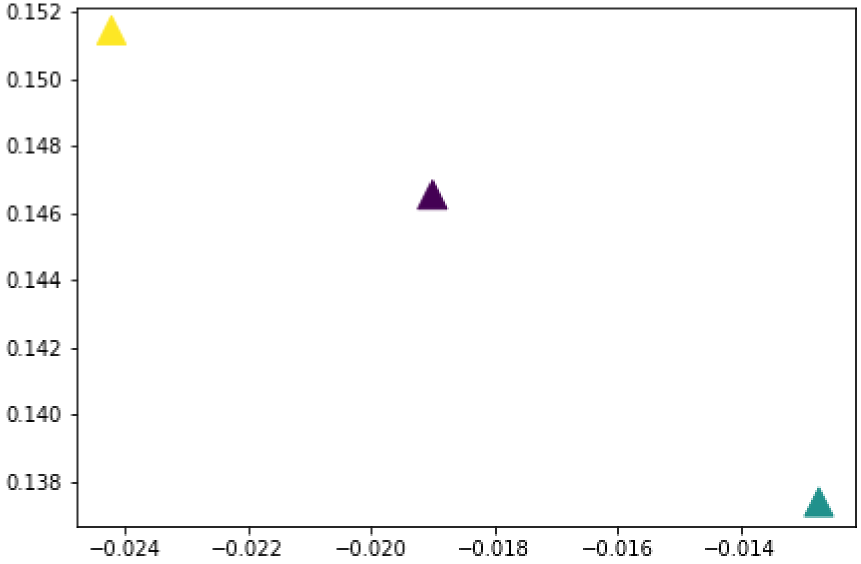
```
n_clusters=3
```

```
kmeans  =  KMeans(n_clusters=n_clusters,  init='k-means++',  max_iter=300,  n_init=500,  random_state=42)
```

```
kmeans.fit(node2vec_output)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=500, n_jobs=None, precompute_distances='auto',
       random_state=42, tol=0.0001, verbose=0)
```

```
t = np.arange(n_clusters)
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=200, c=t,marker="^")
plt.subplots_adjust(left=0.1, bottom=0.1, right=1, top=1)
```



```
subsample=[]
for i in range(kmeans.n_clusters):
    temp = []
    temp=node2vec_output.iloc[kmeans.labels_==i,:]
    subsample.append(temp)


for list in range(len(subsample)):
    print("Group",list+1)
    print(subsample[list])
    print("------------------------------------------------------------")
```

```
Group 1
                   0         1         2  ...        97        98        99
Naeun Son -0.019494  0.148329 -0.005634  ...  0.151387 -0.045348  0.280497
NCT       -0.016736  0.145054 -0.002529  ...  0.149313 -0.044789  0.282389
GFriend   -0.013972  0.142558 -0.010612  ...  0.144197 -0.047642  0.274798
Solar     -0.022353  0.150053 -0.003784  ...  0.151409 -0.043350  0.283544
```

```
Seolhyun  -0.022449  0.146724 -0.011177  ...  0.145636 -0.044387  0.281333

[5 rows x 100 columns]
--------------------------------------------------------------
Group 2
                 0         1         2  ...        97        98        99
BlackPink -0.013286  0.138279 -0.006733  ...  0.134561 -0.043479  0.259630
BTS       -0.012165  0.136504 -0.001157  ...  0.138862 -0.037596  0.263194

[2 rows x 100 columns]
--------------------------------------------------------------
Group 3
             0         1         2  ...        97        98      99
iZone -0.024211  0.151449 -0.005476  ...  0.163354 -0.046334  0.3024

[1 rows x 100 columns]
--------------------------------------------------------------
```

✓ 0s completed at 4:48 PM