Building our own classification model

In this notebook, we will create a classification model to help Perpetual Investors get insightful environmental and social information about the pilot companies. The goal is to help them make data-driven decisions about whether they should invest in these companies. The classification model will assign the companies in three different categories regarding their Environmental Actions:

1) Good Forecasting 2) Improving 3) Bad forecasting

Further on the notebook, we will get into more detail about the requried 'rules' for each classification group.

As mentioned, we are doing this classification model for the 52 US pilot stocks. Moreover, because we know how important the industry factor is, we decided to classify the companies within each industry to make them comparable. Therefore, we will have two classification model. One to classify the companies in the Utility industry and another for the Energy industry.

Steps followed to build our own classification model:

1) Data Cleaning 2) Hierarchical Clustering 3) K-Means Clustering 4) Cluster Profiling 5) Assigned the labels based in the final clusters and rules

We will have a Hierarchical Clustering and K-means clustering for each industry. We used Unsupervised Machine Learning to find patterns and similarities within the companies.

Let's start building the model.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# for distance and h-clustering
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from scipy.spatial.distance import pdist, squareform

#Kmeans clustering
from sklearn.cluster import KMeans, DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn import metrics
from sklearn.decomposition import PCA

# sklearn does have some functionality too, but mostly a wrapper to scipy
from sklearn.metrics import pairwise_distances
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('df_merged.csv')
df_ei = pd.read_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Dataset
stocks = pd.read_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Datase
```

In [2]:

```
sectors = pd.read_csv("/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Datas
stocks['Missing_GHG'] = np.where(stocks['GHG Scope 1'].isna(), 1, 0)
df = pd.merge(stocks, sectors, how='inner',on='Ticker')
df.drop(columns='Name', inplace=True)
stocks = df.copy()
stocks['Utility'] = np.where(stocks.Sector == 'Utilities',1,0)
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 780 entries, 0 to 779
Data columns (total 28 columns):
 #   Column                                            Non-Null Count  Dtype  
---  ------                                            --------------  -----  
 0   Year                                              780 non-null    int64  
 1   Ticker                                            780 non-null    object 
 2   Environmental Disclosure Score                    636 non-null    float64
 3   GHG Scope 1                                       398 non-null    float64
 4   Total Energy Consumption                          243 non-null    float64
 5   Change_in_EDS                                     589 non-null    float64
 6   Change_in_GHG                                     377 non-null    float64
 7   Change_in_TEC                                     237 non-null    float64
 8   Company                                           780 non-null    object 
 9   Total_Assets                                      754 non-null    float64
 10  Net_Income                                        754 non-null    float64
 11  Total_Sales                                       754 non-null    float64
 12  Change_in_Sales                                   754 non-null    float64
 13  Change_in_Assets                                  754 non-null    float64
 14  Change_in_NI                                      754 non-null    float64
 15  ROA                                               754 non-null    float64
 16  Profit_Margin                                     754 non-null    float64
 17  Annual_Stock_Return                               364 non-null    float64
 18  Ratio of GHG Emissions to Total Assets            398 non-null    float64
 19  Ratio of GHG Emissions to Total Sales             398 non-null    float64
 20  Ratio of Total Energy Consumption to Total Assets 243 non-null    float64
 21  Ratio of Total Energy Consumption to Total Sales  243 non-null    float64
 22  Profitable                                        780 non-null    bool   
 23  Logarithm_Total_Assets                            754 non-null    float64
 24  Logarithm_Total_Sales                             754 non-null    float64
 25  Missing_GHG                                       780 non-null    int64  
 26  Sector                                            780 non-null    object 
 27  Utility                                           780 non-null    int64  
dtypes: bool(1), float64(21), int64(3), object(3)
memory usage: 171.4+ KB
```

Data Cleaning - version 1

As presented throughout the project, industry plays a crudential factor when try to understand, interpret and predict the company environmental actions. In this version, we decided to impute the missing values using the industry average of that year.

At the end, we decided to not use this version. However, we want to present that it was part of our analysis.

In [3]:

```
stocks_df = stocks.copy()
stocks_df = stocks_df.loc[:, ~stocks_df.columns.isin(['Logarithm_Total_Assets','Logarithm_Total_Sales','Sector'
stocks_df.loc[np.isinf(stocks_df.Annual_Stock_Return),'Annual_Stock_Return'] = np.nan #We can see infinity valu
```

In [4]:

```
stocks_df.describe().T
```

Out[4]:

	count	mean	std	min	25%	50%	75%	max
Year	780.0	2012.000000	4.323266	2005.000000	2008.000000	2012.000000	2016.000000	2.019000e+03
Environmental Disclosure Score	636.0	31.469654	18.263954	1.400000	15.900000	31.750000	45.500000	8.430000e+01
GHG Scope 1	398.0	28308.354774	32606.231283	52.500000	5225.000000	15042.650000	36650.000000	1.455000e+05
Total Energy Consumption	243.0	138491.043210	308233.410807	101.500000	10630.400000	60000.000000	191643.500000	3.773309e+06
Change_in_EDS	589.0	25.285433	105.813056	-65.363128	0.000000	2.725724	19.090909	1.887500e+03
Change_in_GHG	377.0	-0.660034	26.811189	-87.928545	-8.586678	-1.145510	3.606484	2.408636e+02
Change_in_TEC	237.0	122.143943	1228.707439	-98.965689	-8.361044	0.000000	3.196347	1.575025e+04
Total_Assets	754.0	44006.135068	52446.876654	172.517000	17640.800500	30182.100000	47309.500000	3.625970e+05
Net_Income	754.0	1862.769263	5374.043089	-23119.000000	306.810750	827.396500	1830.750000	4.522000e+04
Total_Sales	754.0	27297.776397	56598.313351	27.253000	6639.350000	11485.987500	16794.000000	4.335260e+05
Change_in_Sales	754.0	4.887176	31.978475	-78.151774	-5.295453	1.881040	11.113570	6.470431e+02
Change_in_Assets	754.0	7.192162	21.082076	-66.324707	0.000000	4.803945	9.095253	1.786625e+02
Change_in_NI	754.0	-10.272885	926.832093	-8982.500000	-33.633989	0.021741	24.550031	1.956129e+04
ROA	754.0	3.092603	7.516937	-122.699289	1.917490	2.855311	4.719352	3.541617e+01
Profit_Margin	754.0	6.290280	20.662635	-362.196459	3.837708	8.414402	12.040351	4.675778e+01
Annual_Stock_Return	362.0	10.945134	24.569376	-62.306948	-2.450866	11.257979	21.377972	1.372194e+02
Ratio of GHG Emissions to Total Assets	398.0	590.490234	625.325587	3.041010	158.325338	363.476883	829.690452	4.677381e+03
Ratio of GHG Emissions to Total Sales	398.0	1838.463042	2011.210641	15.105900	282.722003	654.607648	3313.797971	9.899855e+03
Ratio of Total Energy Consumption to Total Assets	243.0	1928.396940	4716.675535	4.694293	312.373762	1027.232793	2140.752073	6.688604e+04
Ratio of Total Energy Consumption to Total Sales	243.0	5655.626268	16864.951584	15.861853	742.879876	1518.557341	6102.365655	2.457061e+05
Utility	780.0	0.538462	0.498838	0.000000	0.000000	1.000000	1.000000	1.000000e+00

Let's impute null values with that year industry average for all the columns that contain null values.

In [5]:

```
null_columns= stocks_df.columns[stocks_df.isnull().any()]
null_columns = null_columns.tolist()

for column in null_columns:
    stocks_df[column] = stocks_df.groupby(['Year','Utility']).transform('mean')[[column]]
```

In [6]:

```
stocks_df.isna().sum()
```

Out[6]:

```
Year                                                   0
Ticker                                                 0
Environmental Disclosure Score                         0
GHG Scope 1                                           28
Total Energy Consumption                              56
Change_in_EDS                                         52
Change_in_GHG                                         80
Change_in_TEC                                        108
Company                                                0
Total_Assets                                           0
Net_Income                                             0
Total_Sales                                            0
Change_in_Sales                                        0
Change_in_Assets                                       0
Change_in_NI                                           0
ROA                                                    0
Profit_Margin                                          0
Annual_Stock_Return                                  416
Ratio of GHG Emissions to Total Assets                28
Ratio of GHG Emissions to Total Sales                 28
Ratio of Total Energy Consumption to Total Assets     56
Ratio of Total Energy Consumption to Total Sales      56
Profitable                                             0
Utility                                                0
dtype: int64
```

We have less null values now. However, we keep seeing a great amout of null values for the Annual Stock Return and Change in Total Energy Consumption. We are going to proceed to drop these columns and not consider them for the clustering analysis.

In [7]:

```
stocks_df = stocks_df.loc[:, ~stocks_df.columns.isin(['Annual_Stock_Return','Change_in_TEC'])]
stocks_df.head()
```

Out[7]:

	Year	Ticker	Environmental Disclosure Score	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Company	Total_Assets	Net_Income	...	Change
0	2005	AEE	16.566667	NaN	NaN	NaN	NaN	AMEREN CORP	24071.876643	576.930357	...	
1	2006	AEE	19.153846	23845.733333	NaN	33.294759	NaN	AMEREN CORP	25461.556893	743.487464	...	
2	2007	AEE	18.677273	29858.580000	342779.0	49.676880	2.352400	AMEREN CORP	25310.003393	849.919000	...	
3	2008	AEE	19.676923	23344.828571	327939.0	14.110217	-1.376797	AMEREN CORP	27335.062393	897.238429	...	
4	2009	AEE	26.074074	18867.840000	164566.8	117.994489	-8.848381	AMEREN CORP	28166.217107	793.816214	...	

5 rows × 22 columns

In [8]:

```
is_NaN = stocks_df.isnull()
row_has_NaN = is_NaN.any(axis=1)
rows_with_NaN = stocks_df[row_has_NaN]
null_columns1= rows_with_NaN.columns[rows_with_NaN.isnull().any()]
null_columns1 = null_columns1.tolist()
null_columns1.append('Year')
rows_with_NaN[null_columns1]
```

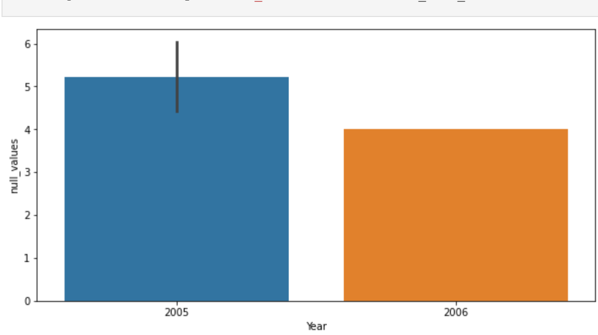
Out[8]:

	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Ratio of GHG Emissions to Total Assets	Ratio of GHG Emissions to Total Sales	Ratio of Total Energy Consumption to Total Assets	Ratio of Total Energy Consumption to Total Sales	Year
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2005
1	23845.733333	NaN	33.294759	NaN	1385.834983	3562.017617	NaN	NaN	2006
15	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2005
16	23845.733333	NaN	33.294759	NaN	1385.834983	3562.017617	NaN	NaN	2006
30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2005
...	...	...	...	...	...	...	...	...	...
721	23845.733333	NaN	33.294759	NaN	1385.834983	3562.017617	NaN	NaN	2006
735	11950.000000	74970.0	NaN	NaN	432.979753	410.866467	2630.710927	1279.43887	2005
750	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2005
751	23845.733333	NaN	33.294759	NaN	1385.834983	3562.017617	NaN	NaN	2006
765	11950.000000	74970.0	NaN	NaN	432.979753	410.866467	2630.710927	1279.43887	2005

80 rows × 9 columns

In [9]:

```
rows_with_NaN['null_values'] = rows_with_NaN.isnull().sum(axis=1)
rows_with_NaN.head()

plt.figure(figsize=(10,5));
sns.barplot(x='Year', y = 'null_values', data = rows_with_NaN);
```



Let's get rid of data from 2005 and 2006 because it is missing data for multiple columns

In [10]:

```
stocks_df = stocks_df.loc[stocks_df.Year >= 2007, ]
stocks_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 676 entries, 2 to 779
Data columns (total 22 columns):
 #   Column                                            Non-Null Count  Dtype  
---  ------                                            --------------  -----  
 0   Year                                              676 non-null    int64  
 1   Ticker                                            676 non-null    object 
 2   Environmental Disclosure Score                    676 non-null    float64
 3   GHG Scope 1                                       676 non-null    float64
 4   Total Energy Consumption                          676 non-null    float64
 5   Change_in_EDS                                     676 non-null    float64
 6   Change_in_GHG                                     676 non-null    float64
 7   Company                                           676 non-null    object 
 8   Total_Assets                                      676 non-null    float64
 9   Net_Income                                        676 non-null    float64
 10  Total_Sales                                       676 non-null    float64
 11  Change_in_Sales                                   676 non-null    float64
 12  Change_in_Assets                                  676 non-null    float64
 13  Change_in_NI                                      676 non-null    float64
 14  ROA                                               676 non-null    float64
 15  Profit_Margin                                     676 non-null    float64
 16  Ratio of GHG Emissions to Total Assets            676 non-null    float64
 17  Ratio of GHG Emissions to Total Sales             676 non-null    float64
 18  Ratio of Total Energy Consumption to Total Assets 676 non-null    float64
 19  Ratio of Total Energy Consumption to Total Sales  676 non-null    float64
 20  Profitable                                        676 non-null    bool   
 21  Utility                                           676 non-null    int64  
dtypes: bool(1), float64(17), int64(2), object(2)
memory usage: 116.8+ KB
```

Hierarchical Clustering - version 1 and utility industry

Now that we have a cleaned dataset, we will start with the clustering. First, we did Hierarchical Clustering for the utility industry.

As a reminder, we are not using this version for the classification model.

In [11]:

```
# 1) Standarized the data
df = stocks_df[stocks_df.Utility == 1]
stock_number = df.select_dtypes('number')
sc = StandardScaler()
stock_scaled = sc.fit_transform(stock_number)
stock_scaled = pd.DataFrame(stock_scaled, columns = stock_number.columns)
stock_scaled
```

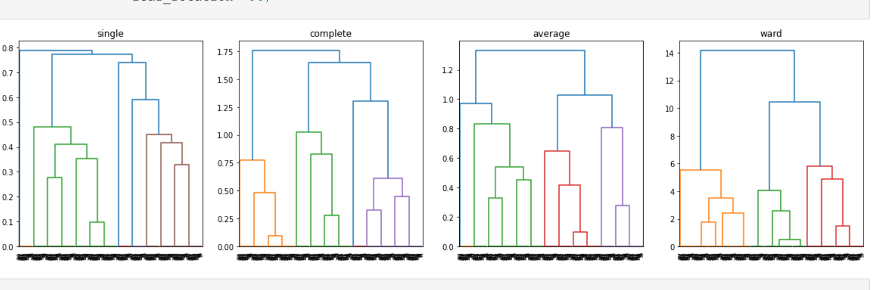
Out[11]:

	Year	Environmental Disclosure Score	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Total_Assets	Net_Income	Total_Sales	Change_in_Sale
0	-1.603567	-1.626588	-0.009989	0.715048	0.879926	0.668179	-1.458084	-0.330097	-1.326754	0.15904
1	-1.336306	-1.514224	-1.193930	0.629004	-0.332286	0.104514	-1.222410	-0.136021	-0.150339	1.23926
2	-1.069045	-0.795163	-2.007669	-0.318249	3.208382	-1.024811	-1.125682	-0.560196	-1.738446	-2.12406
3	-0.801784	-0.724807	-1.137553	-0.395566	-0.466368	0.551322	-0.925449	-0.185963	-1.045936	0.49535
4	-0.534522	-0.552455	0.729316	0.091868	-0.289423	-0.024700	-0.641432	0.017768	-0.567129	0.23404
...	...	...	...	...	...	...	...	...	...	...
359	0.534522	0.528282	0.776012	-0.483558	-0.065203	-1.027699	0.385752	-0.817127	0.425876	-1.60318
360	0.801784	0.877982	0.295243	-0.648703	-0.146088	-0.936210	0.808715	-1.260166	0.253713	-0.54483
361	1.069045	1.241836	-0.314184	-0.734232	-0.264208	-0.261941	0.972987	0.291251	0.707892	0.07931
362	1.336306	1.275571	0.025056	-0.657463	-0.711375	-0.243917	1.285979	1.458217	1.261293	0.71362
363	1.603567	1.507336	-0.410783	-0.770496	-0.480345	-1.032557	1.830051	2.622902	1.319513	-0.17835

364 rows × 19 columns

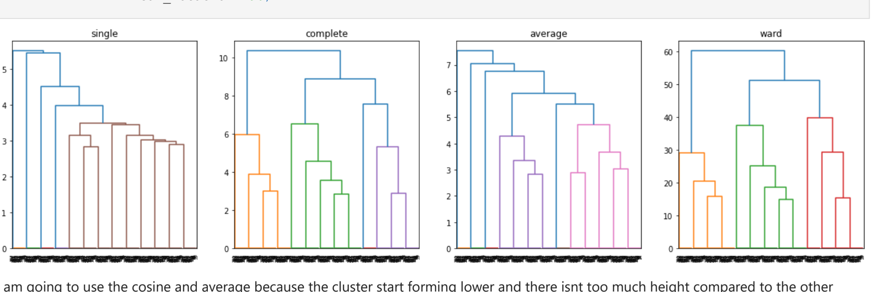
In [12]:

```
#2) Clustering using euclidean and cosine for distance matrix

dc1 = pdist(stock_scaled.values) #euclidean
dc2 = pdist(stock_scaled.values, metric='cosine')

#See now with linkage method and cosine distance matrix work
METHODS = ['single', 'complete', 'average', 'ward']

plt.figure(figsize=(20,5))

for i,m in enumerate(METHODS):
    plt.subplot(1,4,i+1)
    plt.title(m)
    dendrogram(linkage(dc2, method=m),
               leaf_rotation= 90)
```



In [13]:

```
#3 ) See how linkage method and euclidean distance metric work
plt.figure(figsize=(20,5))

for i,m in enumerate(METHODS):
    plt.subplot(1,4,i+1)
    plt.title(m)
    dendrogram(linkage(dc1, method=m),
               leaf_rotation= 90)
```

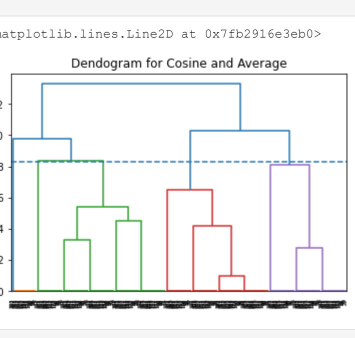


I am going to use the cosine and average because the cluster start forming lower and there isnt too much height compared to the other methods. Clusters are more group together and compact

In [14]:

```
#4) Create the labels
hc1 = linkage(dc2, method='average')
plt.title('Dendogram for Cosine and Average')
dendrogram(hc1,
           leaf_rotation= 90)
plt.axhline(linestyle='--', y=.83)
```

Out[14]: <matplotlib.lines.Line2D at 0x7fb2916e3eb0>



In [15]:

```
labels = fcluster(hc1, 3, criterion='maxclust')
np.unique(labels)
```

Out[15]: array([1, 2, 3], dtype=int32)

In [16]:

```
#put the labels into the clean dataset
df['cluster'] = labels
df.head(3) #Review the dataset with the labels
```

Out[16]:

	Year	Ticker	Environmental Disclosure Score	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Company	Total_Assets	Net_Income	...	Change
2	2007	AEE	18.677273	29858.580000	342779.0	49.676880	2.352400	AMEREN CORP	25310.003393	849.919000	...	20.6
3	2008	AEE	19.676923	23344.828571	327939.0	14.110217	-1.376797	AMEREN CORP	27335.062393	897.238429	...	-56.0
4	2009	AEE	26.074074	18867.840000	164566.8	117.994489	-8.848381	AMEREN CORP	28166.217107	793.816214	...	-8.7

3 rows × 23 columns

In [17]:

```
#How many stocks per cluster
df.cluster.value_counts(dropna=False, sort=False)
```

Out[17]:

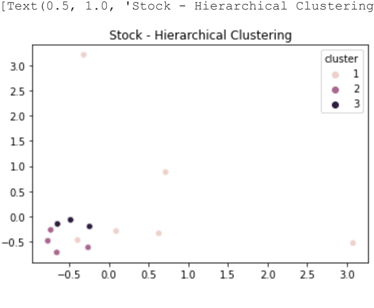
```
1    168
2    112
3     84
Name: cluster, dtype: int64
```

When using three clusters... we can see that the number of observations are more balanced.

In [18]:

```
X = stock_scaled.values
sns.scatterplot(X[:,3],X[:,4],hue=df.cluster, cmap="rainbow").set(title='Stock - Hierarchical Clustering')
```

Out[18]: [Text(0.5, 1.0, 'Stock - Hierarchical Clustering')]

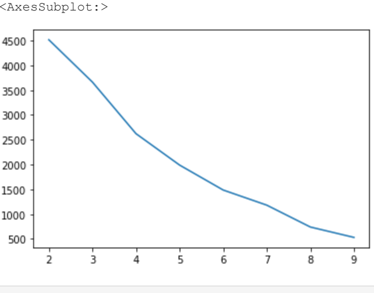


K-Means Clustering - version 1 and utility

In [19]:

```
#Cluster Evaluation - Deciding how many clusters
X = stock_scaled.values
KRANGE = range(2,10)
sse = []

## loop over and evaluate
for k in KRANGE:
  km = KMeans(k)
  labs = km.fit_predict(stock_scaled)
  sse.append(km.inertia_)

#Elbow Method
sns.lineplot(KRANGE,sse)
```

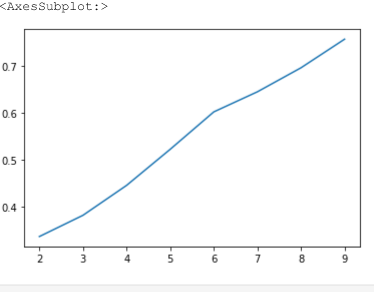
Out[19]: <AxesSubplot:>



In [20]:

```
# Testing K
ss1 = []

for k in KRANGE:
  km = KMeans(k)
  lab = km.fit_predict(stock_scaled)
  ss1.append(metrics.silhouette_score(stock_scaled, lab))

sns.lineplot(KRANGE, ss1)
```

Out[20]: <AxesSubplot:>



In [21]:

```
#It shows 3 clusters are the best option... because it radically descreases when 4.

##K Means for 3
k3 = KMeans(3)
k3.fit(X)
k3_labs = k3.predict(X)
```

In [22]:

```
df['k3'] = k3_labs
df.k3.value_counts(dropna=False, sort=False) #
```

Out[22]:

```
0     84
1    112
2    168
Name: k3, dtype: int64
```

Hierarchical Clustering - version 1 and Energy industry

In [23]:

```
# 1) Standarized the data
df1 = stocks_df[stocks_df.Utility == 0]
stock_number = df1.select_dtypes('number')
sc = StandardScaler()
stock_scaled = sc.fit_transform(stock_number)
stock_scaled = pd.DataFrame(stock_scaled, columns = stock_number.columns)
stock_scaled
```

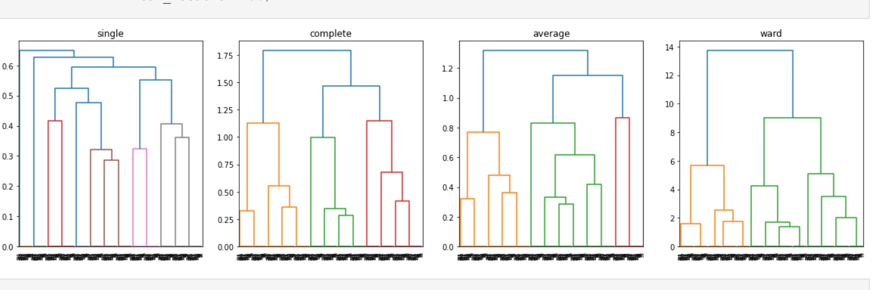
Out[23]:

	Year	Environmental Disclosure Score	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Total_Assets	Net_Income	Total_Sales	Change_in_Sale
0	-1.603567	-1.180250	2.444606	0.382461	0.318353	-0.263506	-1.404091	1.184593	0.380220	0.02944
1	-1.336306	-0.983208	1.642062	3.386287	2.049795	-0.406269	-1.690471	0.365668	1.450673	0.95267
2	-1.069045	-0.778242	0.418996	-0.190871	-0.346212	-2.034185	-1.619526	-0.384349	-0.570848	-1.74549
3	-0.801784	-0.821406	0.304061	-0.254163	-0.784159	-1.471398	-0.776502	0.445092	0.411619	0.70397
4	-0.534522	-0.372013	-0.061599	-0.247434	-0.018908	1.437836	-0.150424	1.210522	1.532144	0.98916
...	...	...	...	...	...	...	...	...	...	...
307	0.534522	0.178634	-0.318692	-0.370579	-0.282827	-0.167982	0.497317	-1.912827	-1.186175	-1.96614
308	0.801784	0.685560	-0.541348	-0.381446	2.085182	-0.633487	0.344429	-1.684325	-1.698863	-1.07714
309	1.069045	1.083076	-1.029470	-0.432682	0.057406	-0.027990	0.332229	-0.570472	-1.194253	1.19626
310	1.336306	1.386493	-1.093107	-0.495630	-0.987165	1.327237	0.587770	0.079518	-0.521884	0.91409
311	1.603567	2.189313	-1.092126	-0.465289	-0.312347	1.060869	0.978736	-1.000412	-0.655745	-0.36867

312 rows × 19 columns

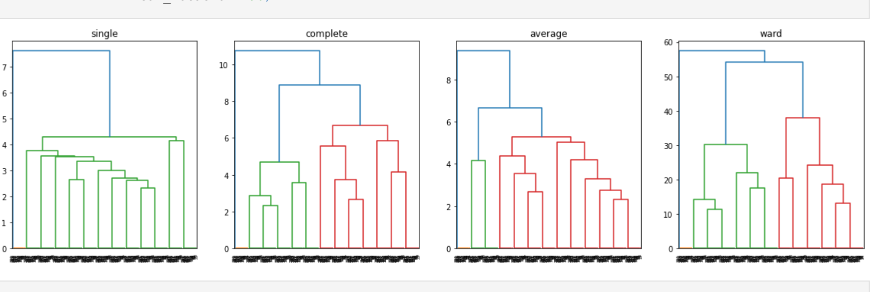
In [24]:

```
#2) Clustering using euclidean and cosine for distance matrix

dc1 = pdist(stock_scaled.values) #euclidean
dc2 = pdist(stock_scaled.values, metric='cosine')

#See now with linkage method and cosine distance matrix work
METHODS = ['single', 'complete', 'average', 'ward']

plt.figure(figsize=(20,5))

for i,m in enumerate(METHODS):
    plt.subplot(1,4,i+1)
    plt.title(m)
    dendrogram(linkage(dc2, method=m),
               leaf_rotation= 90)
```



In [25]:

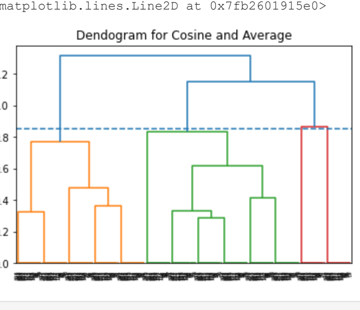
```
#3 ) See how linkage method and euclidean distance metric work
plt.figure(figsize=(20,5))

for i,m in enumerate(METHODS):
    plt.subplot(1,4,i+1)
    plt.title(m)
    dendrogram(linkage(dc1, method=m),
               leaf_rotation= 90)
```



In [26]:

```
#4) Create the labels
hc1 = linkage(dc2, method='average')
plt.title('Dendogram for Cosine and Average')
dendrogram(hc1,
           leaf_rotation= 90)
plt.axhline(linestyle='--', y=.85)
```

Out[26]: <matplotlib.lines.Line2D at 0x7fb2601915e0>



In [27]:

```
labels = fcluster(hc1, 2, criterion='maxclust')
np.unique(labels)
```

Out[27]: array([1, 2], dtype=int32)

In [28]:

```
#put the labels into the clean dataset
df1['cluster'] = labels
df1.head(3) #Review the dataset with the labels
```

Out[28]:

	Year	Ticker	Environmental Disclosure Score	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Company	Total_Assets	Net_Income	...	Chan
47	2007	APA	21.006667	47991.866667	201411.920000	36.438206	-1.411253	APACHE CORP	47428.339600	5389.921400	...	3.
48	2008	APA	22.381250	41021.480000	633597.340000	72.142628	-2.624571	APACHE CORP	45742.297850	3649.938250	...	-7.
49	2009	APA	23.811111	30398.720000	118921.842857	22.734085	-16.460037	APACHE CORP	46159.977238	2056.365095	...	-5

3 rows × 23 columns

In [29]:

```
#How many stocks per cluster
df1.cluster.value_counts(dropna=False, sort=False)
```

Out[29]:

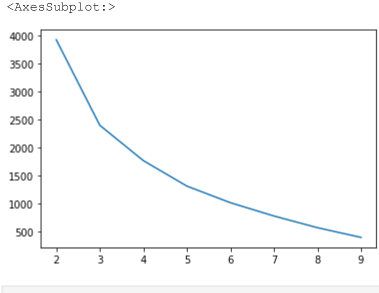
```
1    120
2    192
Name: cluster, dtype: int64
```

K-Means version 1 and energy industry

In [30]:

```
#Cluster Evaluation - Deciding how many clusters
X = stock_scaled.values
KRANGE = range(2,10)
sse = []

## loop over and evaluate
for k in KRANGE:
  km = KMeans(k)
  labs = km.fit_predict(stock_scaled)
  sse.append(km.inertia_)

#Elbow Method
sns.lineplot(KRANGE,sse)
```

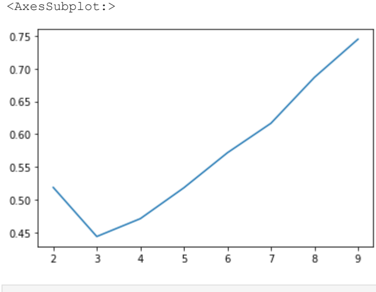
Out[30]: <AxesSubplot:>



In [31]:

```
# Testing K
ss1 = []

for k in KRANGE:
  km = KMeans(k)
  lab = km.fit_predict(stock_scaled)
  ss1.append(metrics.silhouette_score(stock_scaled, lab))

sns.lineplot(KRANGE, ss1)
```

Out[31]: <AxesSubplot:>



In [32]:

```
#It shows 2 clusters are the best option... because it radically descreases when 4.

##K Means for 2
k2 = KMeans(2)
k2.fit(X)
k2_labs = k2.predict(X)
```

In [33]:

```
df1['k2'] = k2_labs
df1.k2.value_counts(dropna=False, sort=False) #
```

Out[33]:

```
0    120
1    192
Name: k2, dtype: int64
```

Cluster profiling for Utility industry

In [34]:

```
df_clus = df.drop(columns=['k3','Utility'])
stock_numeric = df_clus.select_dtypes('number')
stock_numeric
```

Out[34]:

	Year	Environmental Disclosure Score	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Total_Assets	Net_Income	Total_Sales	Change_in_Sa
2	2007	18.677273	29858.580000	342779.000000	49.676880	2.352400	25310.003393	849.919000	9606.342643	2.646
3	2008	19.676923	23344.828571	327939.000000	14.110217	-1.376797	27335.062393	897.238429	10370.114714	8.094
4	2009	26.074074	18867.840000	164566.800000	117.994489	-8.848381	28166.217107	793.816214	9339.057071	-8.867
5	2010	26.700000	23655.000000	151232.066667	10.176206	1.579278	29886.748250	885.061679	9788.659786	4.342
6	2011	28.233333	33926.050000	235299.600000	15.367831	-2.231674	32327.204107	934.735250	10099.519286	3.024
...	...	...	...	...	...	...	...	...	...	...
760	2015	37.848148	34182.959091	136056.161538	21.946524	-8.867489	41153.442643	731.171321	10744.214536	-6.240
761	2016	40.959259	31537.887500	107573.650000	19.573316	-8.262200	44787.810714	623.149750	10632.439714	-0.903
762	2017	44.196296	28184.976000	92822.446667	16.107659	-3.801250	46199.345750	1001.415929	10927.309536	2.244
763	2018	44.496429	30051.388889	106062.837500	2.987642	-3.682008	48888.773607	1285.945214	11286.598071	5.443
764	2019	46.558333	27653.507692	86567.993750	9.766126	-8.899627	53563.792607	1569.918679	11324.396429	0.944

364 rows × 19 columns

In [35]:

```
clus_profile = stock_numeric.groupby("cluster").mean()
clus_profile
```

Out[35]:

	Year	Environmental Disclosure Score	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Total_Assets	Net_Income	Total_Sales	Ch
cluster										
1	2009.666667	25.344218	27882.777540	328710.813889	35.961816	-1.020104	30076.236827	868.060476	9959.142786	
2	2017.000000	42.386839	31232.514198	114918.547257	8.697172	-4.941242	47201.197938	1211.610027	11239.294679	
3	2014.333333	36.438272	32216.413308	140325.737179	19.871690	-0.333919	40880.757940	680.145893	10456.108976	

In [36]:
```
scp = StandardScaler()
cluster_scaled = scp.fit_transform(clus_profile)
cluster_scaled = pd.DataFrame(cluster_scaled, index=clus_profile.index, columns=clus_profile.columns)
plt.figure(figsize=(9,9))
sns.heatmap(cluster_scaled.T, cmap="Blues", center=0)
```

Out[36]: <AxesSubplot:xlabel='cluster'>

Cluster profiling for Energy industry

In [37]:
```
df_clus = df1.drop(columns=['cluster','Utility'])
stock_numeric = df_clus.select_dtypes('number')
stock_numeric
```

Out[37]:

	Year	Environmental Disclosure Score	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Total_Assets	Net_Income	Total_Sales	Change_in_S
47	2007	21.006667	47991.866667	201411.920000	36.438206	-1.411253	47428.339600	5389.921400	53283.519150	7.14
48	2008	22.381250	41021.480000	633597.340000	72.142628	-2.624571	45742.297850	3649.938250	65747.774900	25.25
49	2009	23.811111	30398.720000	118921.842857	22.734085	-16.460037	46159.977238	2056.365095	42209.385286	-27.67
50	2010	23.510000	29400.470000	109815.550000	13.703086	-11.676973	51123.226304	3818.690130	53649.131391	20.37
51	2011	26.645000	26224.576923	110783.700000	29.483484	13.048245	54809.214130	5445.012609	66696.417739	25.97
...	...	...	...	...	...	...	...	...	...	
775	2015	30.486364	23991.630769	93065.791667	24.041154	-0.599407	58622.744870	-1191.216609	35044.578043	-32.00
776	2016	34.022727	22057.786667	91502.346154	72.872346	-4.555677	57722.630217	-705.714348	29074.887957	-14.56
777	2017	36.795833	17818.272222	84130.535714	31.057173	0.590373	57650.799917	1660.907042	34950.510083	30.03
778	2018	38.912500	17265.568421	75073.717647	9.516871	12.108277	59155.281750	3041.950750	42779.515333	24.49
779	2019	44.513043	17274.085714	79439.025000	23.432416	9.844451	61457.067292	747.406042	41220.846833	-0.66

312 rows × 19 columns

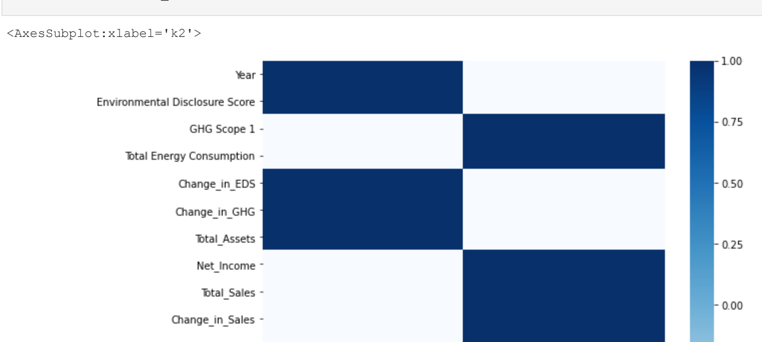
In [38]:
```
clus_profile = stock_numeric.groupby("k2").mean()
clus_profile
```

Out[38]:

	Year	Environmental Disclosure Score	GHG Scope 1	Total Energy Consumption	Change_in_EDS	Change_in_GHG	Total_Assets	Net_Income	Total_Sales	Change_in_S
k2										
0	2017.0	36.946094	19681.468759	84642.283236	32.183992	3.477604	58921.704809	710.666575	36614.067650	1.457
1	2010.5	24.424015	31183.412684	184972.722390	28.429270	-0.827587	53678.026211	4224.453474	56507.662232	9.756

In [39]:
```
scp = StandardScaler()
cluster_scaled = scp.fit_transform(clus_profile)
cluster_scaled = pd.DataFrame(cluster_scaled, index=clus_profile.index, columns=clus_profile.columns)
plt.figure(figsize=(9,9))
sns.heatmap(cluster_scaled.T, cmap="Blues", center=0)
```

Out[39]: <AxesSubplot:xlabel='k2'>



Data Cleaning - version 2

We decided to not consider the first data cleaning version for the cluster analysis. Our second and final version consists and just using the companies that are not missing values for the GHG Scope 1. We decided to do this to be consistent with the predictive models that were built in Sprint #4. As a reminder, in that sprint, we build multiple models to predict the GHG Scope 1 and we only used the observations that didn't have null values.

After subsetting for the companies that have GHG Scope data, we collected the unique tickers and calculate the average for the features that we are insured to cluster by. For this version, we are going to consider the features GHG Scope 1, Total Assets, and Total Sales. Our goal is to be able to find patterns, similarities, and groups for these stocks.

In [40]:
```
stocks_clean = stocks.copy()
stocks_clean['time_trend'] = stocks_clean.groupby('Ticker').cumcount()
stocks_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 780 entries, 0 to 779
Data columns (total 29 columns):
 #   Column                                             Non-Null Count  Dtype
---  ------                                             --------------  -----
 0   Year                                               780 non-null    int64
 1   Ticker                                             780 non-null    object
 2   Environmental Disclosure Score                     636 non-null    float64
 3   GHG Scope 1                                        398 non-null    float64
 4   Total Energy Consumption                           243 non-null    float64
 5   Change_in_EDS                                      589 non-null    float64
 6   Change_in_GHG                                      377 non-null    float64
 7   Change_in_TEC                                      237 non-null    float64
 8   Company                                            780 non-null    object
 9   Total_Assets                                       754 non-null    float64
 10  Net_Income                                         754 non-null    float64
 11  Total_Sales                                        754 non-null    float64
 12  Change_in_Sales                                    754 non-null    float64
 13  Change_in_Assets                                   754 non-null    float64
 14  Change_in_NI                                       754 non-null    float64
 15  ROA                                                754 non-null    float64
 16  Profit_Margin                                      754 non-null    float64
 17  Annual_Stock_Return                                364 non-null    float64
 18  Ratio of GHG Emissions to Total Assets             398 non-null    float64
 19  Ratio of GHG Emissions to Total Sales              398 non-null    float64
 20  Ratio of Total Energy Consumption to Total Assets  243 non-null    float64
 21  Ratio of Total Energy Consumption to Total Sales   243 non-null    float64
 22  Profitable                                         780 non-null    bool
 23  Logarithm_Total_Assets                             754 non-null    float64
 24  Logarithm_Total_Sales                              754 non-null    float64
 25  Missing_GHG                                        780 non-null    int64
 26  Sector                                             780 non-null    object
 27  Utility                                            780 non-null    int64
 28  time_trend                                         780 non-null    int64
dtypes: bool(1), float64(21), int64(4), object(3)
memory usage: 177.5+ KB
```

In [41]:
```
stocks_clean = stocks_clean.loc[stocks_clean['Missing_GHG'] == 0, stocks_clean.columns.isin(['Company','GHG Sco
stocks_clean
```

Out[41]:

	GHG Scope 1	Company	Total_Assets	Total_Sales	Utility
8	32978.0	AMEREN CORP	21042.0	5838.0	1
9	30675.0	AMEREN CORP	22676.0	6053.0	1
10	28548.2	AMEREN CORP	23640.0	6098.0	1
11	26795.6	AMEREN CORP	24699.0	6076.0	1
12	30200.4	AMEREN CORP	25945.0	6177.0	1
...	...	...	...	...	...
775	118000.0	EXXON MOBIL CORP	336758.0	236810.0	0
776	120000.0	EXXON MOBIL CORP	330314.0	197518.0	0
777	117000.0	EXXON MOBIL CORP	348691.0	237162.0	0
778	119000.0	EXXON MOBIL CORP	346196.0	279332.0	0
779	114000.0	EXXON MOBIL CORP	362597.0	255583.0	0

398 rows × 5 columns

In [42]:
```
agg = stocks_clean.groupby('Company')[['GHG Scope 1','Total_Assets','Total_Sales']].mean().reset_index()
stocks_cleaned = pd.merge(agg, stocks_clean, on='Company', how = 'inner',suffixes=('', '_drop'))
stocks_cleaned.drop([col for col in stocks_cleaned.columns if 'drop' in col], axis=1, inplace=True)
stocks_cleaned.drop_duplicates(inplace=True)
stocks_cleaned.head()
```

Out[42]:

	Company	GHG Scope 1	Total_Assets	Total_Sales	Utility
0	AES CORP (THE)	66456.888889	37643.333333	14251.555556	1
9	ALLIANT ENERGY CORP	15811.300000	14436.700000	3427.600000	1
14	AMEREN CORP	29028.414286	24878.571429	6063.285714	1
21	AMERICAN ELECTRIC POWER CO	96828.785714	64374.571429	16056.042857	1
28	AMERICAN WATER WORKS CO INC	60.855556	15455.169667	2980.634000	1

We decided to include the Total Assets and Total Sales predictors because they can helps us explain some similarities and account for the company size.

In [43]:
```
stocks_cleaned.shape
```

Out[43]: (49, 5)

Hierarchical Clustering - Version 2 and utility industry

We follow the same steps as with version 1.

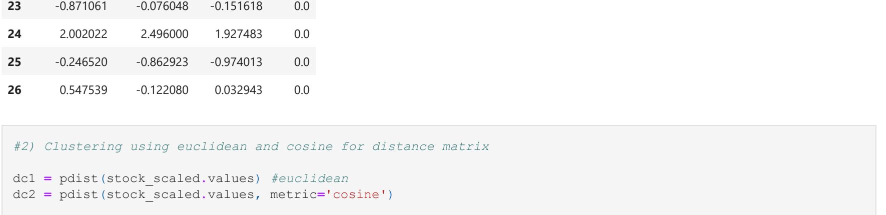
1) Standarized data 2) Clustering using euclidean and cosine for distance matrix 3) Try different linkage method for each distance matrix 4) Create labels and add them to the original dataset 5) Sort by cluster values

In [44]:
```
# 1) Standarized the data
stocks_clean1 = stocks_cleaned[stocks_cleaned.Utility == 1]
stock_number = stocks_clean1.select_dtypes('number')
sc = StandardScaler()
stock_scaled = sc.fit_transform(stock_number)
stock_scaled = pd.DataFrame(stock_scaled, columns = stock_number.columns)
stock_scaled
```

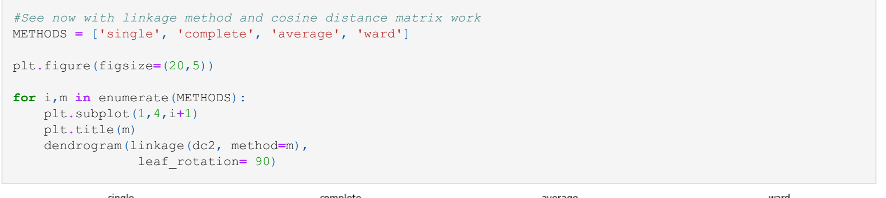
Out[44]:

	GHG Scope 1	Total_Assets	Total_Sales	Utility
0	1.126271	-0.254604	0.532308	0.0
1	-0.582903	-1.050878	-1.308093	0.0
2	-0.136856	-0.692593	-0.859946	0.0
3	2.151255	0.662608	0.839126	0.0
4	-1.114445	-1.015932	-1.384091	0.0
5	-0.919947	-0.571947	-0.048667	0.0
6	-0.683731	-0.722413	-0.755936	0.0
7	-1.007818	-0.019635	0.247305	0.0
8	0.283401	1.121406	0.358919	0.0
9	0.097059	-0.501885	-0.017759	0.0
10	2.663225	2.677734	1.925537	0.0
11	-1.001325	0.224934	0.212443	0.0
12	0.023720	-0.075010	0.038954	0.0
13	-0.167474	-0.661423	-1.083593	0.0
14	-1.078703	-0.413518	-0.580860	0.0
15	-0.719926	1.490205	2.676018	0.0
16	0.060441	-0.107098	0.271472	0.0
17	0.365383	1.457666	0.939691	0.0
18	-0.489958	-0.843760	-0.863678	0.0
19	1.023567	-0.621679	-0.045009	0.0
20	-0.653269	-1.047852	-1.307981	0.0
21	-0.141989	-0.117668	-0.592299	0.0
22	-0.527957	-0.351606	-0.028656	0.0
23	-0.871061	-0.076048	-0.151618	0.0
24	2.002022	2.496000	1.927483	0.0
25	-0.246520	-0.862923	-0.974013	0.0
26	0.547539	-0.122080	0.032943	0.0

In [45]:
```
#2) Clustering using euclidean and cosine for distance matrix

dc1 = pdist(stock_scaled.values) #euclidean
dc2 = pdist(stock_scaled.values, metric='cosine')

#See now with linkage method and cosine distance matrix work
METHODS = ['single', 'complete', 'average', 'ward']

plt.figure(figsize=(20,5))

for i,m in enumerate(METHODS):
    plt.subplot(1,4,i+1)
    plt.title(m)
    dendrogram(linkage(dc2, method=m),
               leaf_rotation= 90)
```

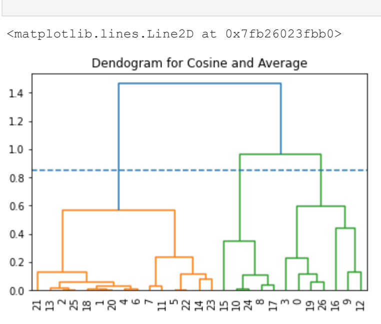


In [46]:
```
#3 ) See how linkage method and euclidean distance metric work
plt.figure(figsize=(20,5))

for i,m in enumerate(METHODS):
    plt.subplot(1,4,i+1)
    plt.title(m)
    dendrogram(linkage(dc1, method=m),
               leaf_rotation= 90)
```



In [47]:
```
#4) Create the labels
hc1 = linkage(dc2, method='average')
plt.title('Dendogram for Cosine and Average')
dendrogram(hc1,
           leaf_rotation= 90)
plt.axhline(linestyle='--', y=.85)
```

Out[47]: <matplotlib.lines.Line2D at 0x7fb26023fbb0>



In [48]:
```
labels = fcluster(hc1, 3, criterion='maxclust')
np.unique(labels)
```

Out[48]: array([1, 2, 3], dtype=int32)

In [49]:
```
#put the labels into the clean dataset
stocks_clean1['cluster'] = labels
stocks_clean1.head(3) #Review the dataset with the labels
```

Out[49]:

	Company	GHG Scope 1	Total_Assets	Total_Sales	Utility	cluster
0	AES CORP (THE)	66456.888889	37643.333333	14251.555556	1	3
9	ALLIANT ENERGY CORP	15811.300000	14436.700000	3427.600000	1	1
14	AMEREN CORP	29028.414286	24878.571429	6063.285714	1	1

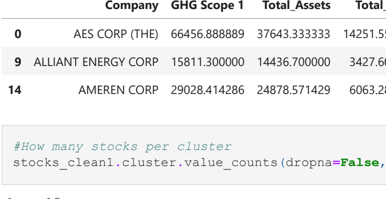
In [50]:
```
#How many stocks per cluster
stocks_clean1.cluster.value_counts(dropna=False, sort=False)
```

Out[50]:
```
1    15
2     5
3     7
Name: cluster, dtype: int64
```

K-Means Clustering - Version 2 and utility industry

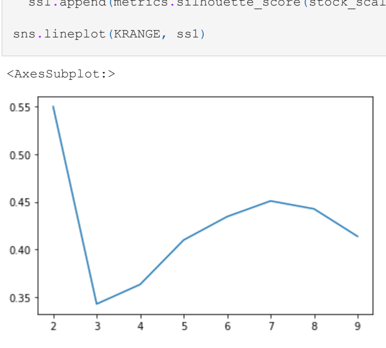
In [51]:
```
#Cluster Evaluation - Deciding how many clusters
X = stock_scaled.values
KRANGE = range(2,10)
sse = []

## loop over and evaluate
for k in KRANGE:
  km = KMeans(k)
  labs = km.fit_predict(stock_scaled)
  sse.append(km.inertia_)

#Elbow Method
sns.lineplot(KRANGE,sse)
```

Out[51]: <AxesSubplot:>



In [52]:
```
# Testing K
ss1 = []

for k in KRANGE:
  km = KMeans(k)
  lab = km.fit_predict(stock_scaled)
  ss1.append(metrics.silhouette_score(stock_scaled, lab))

sns.lineplot(KRANGE, ss1)
```

Out[52]: <AxesSubplot:>



In [53]:
```
##K Means for 3
k3 = KMeans(3)
k3.fit(X)
k3_labs = k3.predict(X)
```

In [54]:
```
stocks_clean1['k3'] = k3_labs
stocks_clean1.k3.value_counts(dropna=False, sort=False) #
```

Out[54]:
```
0     5
1     9
2    13
Name: k3, dtype: int64
```

We will proceed to export this utility dataset which will be used later on this notebook and in other ones.

In [55]:
```
stocks_clean1.to_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Datase
```

Hierarchical Clustering - Version 2 and energy industry

We follow the same steps as with version 2 and utility industry.

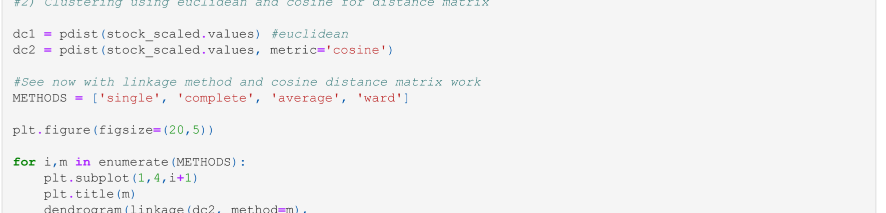
1) Standarized data 2) Clustering using euclidean and cosine for distance matrix 3) Try different linkage method for each distance matrix 4) Create labels and add them to the original dataset 5) Sort by cluster values

In [56]:
```
# 1) Standarized the data
stocks_clean2 = stocks_cleaned[stocks_cleaned.Utility == 0]
stock_number = stocks_clean2.select_dtypes('number')
sc = StandardScaler()
stock_scaled = sc.fit_transform(stock_number)
stock_scaled = pd.DataFrame(stock_scaled, columns = stock_number.columns)
stock_scaled
```

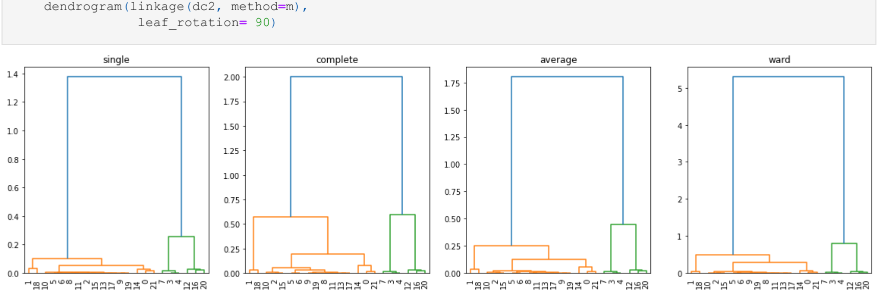
Out[56]:

	GHG Scope 1	Total_Assets	Total_Sales	Utility
0	-0.097826	-0.322949	-0.559181	0.0
1	-0.610522	-0.063158	-0.416030	0.0
2	-0.604975	-0.777642	-0.674909	0.0
3	1.647815	2.258538	1.715311	0.0
4	0.682291	0.900561	0.664007	0.0
5	-0.429168	-0.359049	-0.541567	0.0
6	-0.459472	-0.359508	-0.499440	0.0
7	3.858739	3.575635	3.667599	0.0
8	-0.463215	-0.439673	-0.387283	0.0
9	-0.455794	-0.379913	-0.429915	0.0
10	-0.439587	-0.686757	-0.501088	0.0
11	-0.360346	-0.356966	-0.302825	0.0
12	0.248116	-0.177570	0.372690	0.0
13	-0.588127	-0.558001	-0.589056	0.0
14	-0.227255	-0.110407	-0.455800	0.0
15	-0.532507	-0.615462	-0.538966	0.0
16	0.368734	-0.074074	0.545168	0.0
17	-0.553239	-0.568460	-0.572683	0.0
18	-0.565953	0.045889	-0.232028	0.0
19	-0.606373	-0.475842	-0.518887	0.0
20	0.368031	-0.215212	0.848223	0.0
21	-0.179366	-0.239979	-0.593339	0.0

In [57]:
```
#2) Clustering using euclidean and cosine for distance matrix

dc1 = pdist(stock_scaled.values) #euclidean
dc2 = pdist(stock_scaled.values, metric='cosine')

#See now with linkage method and cosine distance matrix work
METHODS = ['single', 'complete', 'average', 'ward']

plt.figure(figsize=(20,5))

for i,m in enumerate(METHODS):
    plt.subplot(1,4,i+1)
    plt.title(m)
    dendrogram(linkage(dc2, method=m),
               leaf_rotation= 90)
```

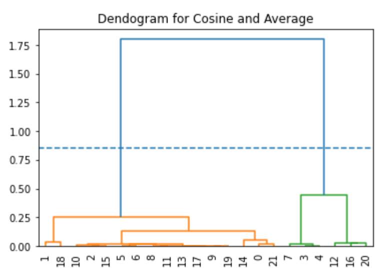


In [58]:
```
#3 ) See how linkage method and euclidean distance metric work
plt.figure(figsize=(20,5))

for i,m in enumerate(METHODS):
    plt.subplot(1,4,i+1)
    plt.title(m)
    dendrogram(linkage(dc1, method=m),
               leaf_rotation= 90)
```



In [59]:
```
#4) Create the labels
hc1 = linkage(dc2, method='average')
plt.title('Dendogram for Cosine and Average')
dendrogram(hc1,
           leaf_rotation= 90)
plt.axhline(linestyle='--', y=.85)
```

Out[59]: <matplotlib.lines.Line2D at 0x7fb27167fee0>



In [60]:
```
labels = fcluster(hc1, 2, criterion='maxclust')
np.unique(labels)
```

Out[60]: array([1, 2], dtype=int32)

In [61]:
```
#put the labels into the clean dataset
stocks_clean2['cluster'] = labels
stocks_clean2.head(3) #Review the dataset with the labels
```

Out[61]:

	Company	GHG Scope 1	Total_Assets	Total_Sales	Utility	cluster
37	APACHE CORP	14952.566667	36178.852333	10675.480500	0	1
49	BAKER HUGHES CO	373.166667	54286.000000	21324.666667	0	1
52	CABOT OIL & GAS CORP	530.900000	4487.245000	2066.277000	0	1

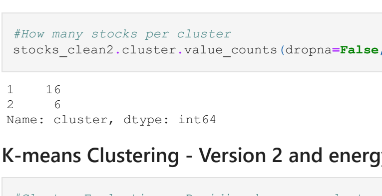
In [62]:
```
#How many stocks per cluster
stocks_clean2.cluster.value_counts(dropna=False, sort=False)
```

Out[62]:
```
1    16
2     6
Name: cluster, dtype: int64
```

K-means Clustering - Version 2 and energy industry

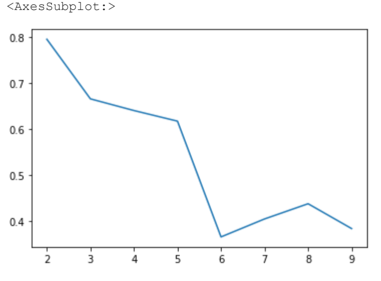
In [63]:
```
#Cluster Evaluation - Deciding how many clusters
X = stock_scaled.values
KRANGE = range(2,10)
sse = []

## loop over and evaluate
for k in KRANGE:
  km = KMeans(k)
  labs = km.fit_predict(stock_scaled)
  sse.append(km.inertia_)

#Elbow Method
sns.lineplot(KRANGE,sse)
```

Out[63]: <AxesSubplot:>



In [64]:
```
# Testing K
ss1 = []

for k in KRANGE:
  km = KMeans(k)
  lab = km.fit_predict(stock_scaled)
  ss1.append(metrics.silhouette_score(stock_scaled, lab))

sns.lineplot(KRANGE, ss1)
```

Out[64]: <AxesSubplot:>



In [65]:
```
k3 = KMeans(3)
k3.fit(X)
k3_labs = k3.predict(X)
```

In [66]:
```
stocks_clean2['k3'] = k3_labs
stocks_clean2['k3'] = stocks_clean2['k3'] + 1
stocks_clean2.k3.value_counts(dropna=False, sort=False) #
```

Out[66]:
```
1    16
2     2
3     4
Name: k3, dtype: int64
```

Not good at all.... We will proceed to export this utility dataset which will be used later on this notebook and in other ones.

In [67]:
```
stocks_clean2.to_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Datase
```

Cluster profiling for version 2 and utility industry

In [68]:
```
df_clus = stocks_clean1.drop(columns=['k3','Utility'])
stock_numeric = df_clus.select_dtypes('number')
stock_numeric
```

Out[68]:

	GHG Scope 1	Total_Assets	Total_Sales	cluster
0	66456.888889	37643.333333	14251.555556	3
9	15811.300000	14436.700000	3427.600000	1
14	29028.414286	24878.571429	6063.285714	1
21	96828.785714	64374.571429	16056.042857	3
28	60.855556	15455.169667	2980.634000	1
53	5824.166667	28394.666667	10834.666667	1
70	12823.625000	24009.500000	6675.000000	1
88	3220.409091	44491.272727	12575.363636	1
110	41481.280000	77745.800000	13231.800000	2
115	35959.666667	30436.555556	11016.444444	3
124	111999.288889	123103.444444	22445.555556	2
133	3412.811111	51619.000000	12370.333333	1
142	33786.523077	42877.418154	11349.992538	3
158	28121.150000	25787.000000	4747.950000	1
160	1119.962500	33011.921875	7704.678250	1
168	11751.090909	88494.090909	26859.363636	2
193	34874.625000	41942.250000	12717.500000	3
245	43910.537500	87545.750000	16647.500000	2
253	18565.423077	20472.946154	6041.338462	1
267	63413.590909	26945.272727	10856.181818	3
310	13726.250000	14524.903667	3428.261833	1
323	28876.300000	41634.200000	7637.400000	1
328	17439.442857	34816.285714	10952.357143	1
353	7272.727273	42847.181818	10229.181818	1
364	92406.800000	117807.000000	22457.000000	2
372	25778.878571	19914.457143	5392.428571	1
391	49308.114286	41505.598000	11314.637571	3

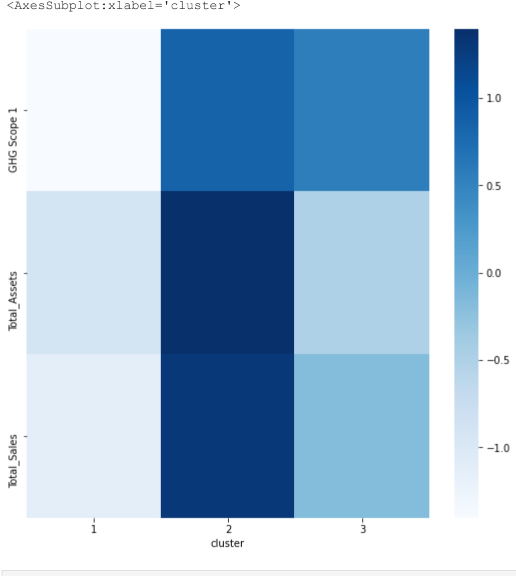
In [69]:
```
clus_profile = stock_numeric.groupby("cluster").mean()
clus_profile
```

Out[69]:

	GHG Scope 1	Total_Assets	Total_Sales
cluster			
1	14072.114399	29086.251791	7404.031962
2	60309.799460	98939.217071	20328.243838
3	54375.456363	40817.857028	12508.907826

In [70]:
```
scp = StandardScaler()
cluster_scaled = scp.fit_transform(clus_profile)
cluster_scaled = pd.DataFrame(cluster_scaled, index=clus_profile.index, columns=clus_profile.columns)
plt.figure(figsize=(9,9))
sns.heatmap(cluster_scaled.T, cmap="Blues", center=0)
```

Out[70]: <AxesSubplot:xlabel='cluster'>



In [71]:
```
df_clus1 = stocks_clean1.drop(columns=['cluster','Utility'])
stock_numeric = df_clus1.select_dtypes('number')
stock_numeric
```

Out[71]:

	GHG Scope 1	Total_Assets	Total_Sales	k3
0	66456.888889	37643.333333	14251.555556	2
9	15811.300000	14436.700000	3427.600000	1
14	29028.414286	24878.571429	6063.285714	1
21	96828.785714	64374.571429	16056.042857	0
28	60.855556	15455.169667	2980.634000	1
53	5824.166667	28394.666667	10834.666667	2
70	12823.625000	24009.500000	6675.000000	1
88	3220.409091	44491.272727	12575.363636	2
110	41481.280000	77745.800000	13231.800000	2
115	35959.666667	30436.555556	11016.444444	2
124	111999.288889	123103.444444	22445.555556	0
133	3412.811111	51619.000000	12370.333333	2
142	33786.523077	42877.418154	11349.992538	2
158	28121.150000	25787.000000	4747.950000	1
160	1119.962500	33011.921875	7704.678250	1
168	11751.090909	88494.090909	26859.363636	0
193	34874.625000	41942.250000	12717.500000	2
245	43910.537500	87545.750000	16647.500000	0
253	18565.423077	20472.946154	6041.338462	1
267	63413.590909	26945.272727	10856.181818	2
310	13726.250000	14524.903667	3428.261833	1
323	28876.300000	41634.200000	7637.400000	2
328	17439.442857	34816.285714	10952.357143	2
353	7272.727273	42847.181818	10229.181818	2
364	92406.800000	117807.000000	22457.000000	0
372	25778.878571	19914.457143	5392.428571	1
391	49308.114286	41505.598000	11314.637571	2

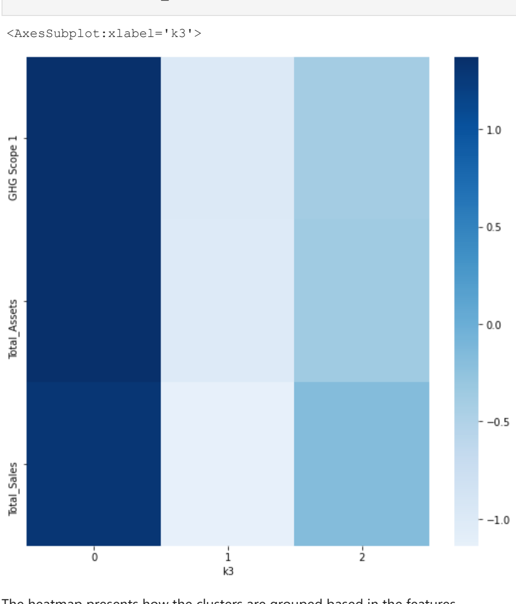
In [72]:
```
clus_profile = stock_numeric.groupby("k3").mean()
clus_profile
```

Out[72]:

k3	GHG Scope 1	Total_Assets	Total_Sales
0	71379.300602	96264.971356	20893.092410
1	16115.095443	21387.907770	5162.352981
2	30102.041987	41761.448823	11487.493425

In [73]:
```
scp = StandardScaler()
cluster_scaled = scp.fit_transform(clus_profile)
cluster_scaled = pd.DataFrame(cluster_scaled, index=clus_profile.index, columns=clus_profile.columns)
plt.figure(figsize=(9,9))
sns.heatmap(cluster_scaled.T, cmap="Blues", center=0)
```

Out[73]: <AxesSubplot:xlabel='k3'>



The heatmap presents how the clusters are grouped based in the features.

Cluster profiling for version 2 and energy industry

Let's do the same thing for the companies in the energy industry

In [74]:
```
df_clus = stocks_clean2.drop(columns=['k3','Utility'])
stock_numeric = df_clus.select_dtypes('number')
stock_numeric
```

Out[74]:

	GHG Scope 1	Total_Assets	Total_Sales	cluster
37	14952.566667	36178.852333	10675.480500	1
49	373.166667	54286.000000	21324.666667	1
52	530.900000	4487.245000	2066.277000	1
56	64592.857143	216105.857143	179877.785714	2
74	37136.521429	121456.214286	101670.000000	2
99	5530.263636	33662.727273	11985.818182	1
155	4668.533333	33630.720000	15119.670000	1
179	127464.285714	307906.142857	325111.000000	2
197	4562.100000	28043.285714	23463.142857	1
204	4773.108333	32208.500000	20291.750000	1
216	5234.000000	10821.814250	14997.060750	1
220	7487.333333	33807.866667	29746.133333	1
235	24790.000000	46311.600000	79998.500000	2
266	1010.000000	19796.000000	8453.000000	1
278	11272.013333	50992.800000	18366.066667	1
293	2591.666667	15790.961583	12179.242167	1
305	28220.000000	53525.200000	92829.400000	2
322	2002.100000	19067.000000	9671.000000	1
342	1640.545455	61886.454545	35012.818182	1
366	491.133333	25522.333333	13672.966667	1
369	28200.000000	43688.000000	115374.000000	2
386	12633.820000	41961.800000	8134.400000	1

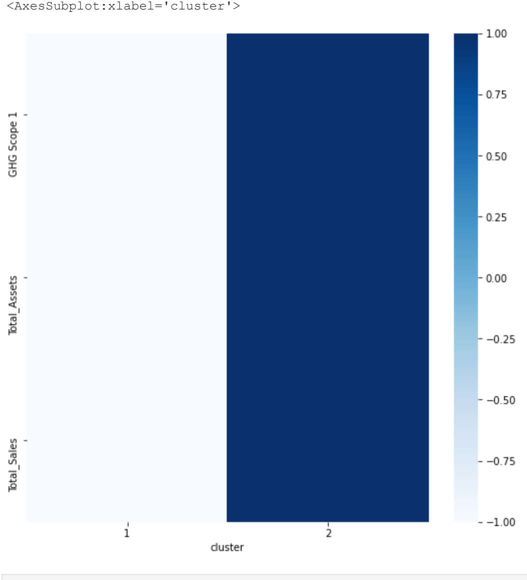
In [75]:
```
clus_profile = stock_numeric.groupby("cluster").mean()
clus_profile
```

Out[75]:

cluster	GHG Scope 1	Total_Assets	Total_Sales
1	4984.578172	31384.022544	15947.468311
2	51733.944048	131498.835714	149143.447619

In [76]:
```
scp = StandardScaler()
cluster_scaled = scp.fit_transform(clus_profile)
cluster_scaled = pd.DataFrame(cluster_scaled, index=clus_profile.index, columns=clus_profile.columns)
plt.figure(figsize=(9,9))
sns.heatmap(cluster_scaled.T, cmap="Blues", center=0)
```

Out[76]: <AxesSubplot:xlabel='cluster'>



In [77]:
```
df_clus = stocks_clean2.drop(columns=['cluster','Utility'])
stock_numeric = df_clus.select_dtypes('number')
stock_numeric
```

Out[77]:

	GHG Scope 1	Total_Assets	Total_Sales	k3
37	14952.566667	36178.852333	10675.480500	1
49	373.166667	54286.000000	21324.666667	1
52	530.900000	4487.245000	2066.277000	1
56	64592.857143	216105.857143	179877.785714	2
74	37136.521429	121456.214286	101670.000000	3
99	5530.263636	33662.727273	11985.818182	1
155	4668.533333	33630.720000	15119.670000	1
179	127464.285714	307906.142857	325111.000000	2
197	4562.100000	28043.285714	23463.142857	1
204	4773.108333	32208.500000	20291.750000	1
216	5234.000000	10821.814250	14997.060750	1
220	7487.333333	33807.866667	29746.133333	1
235	24790.000000	46311.600000	79998.500000	3
266	1010.000000	19796.000000	8453.000000	1
278	11272.013333	50992.800000	18366.066667	1
293	2591.666667	15790.961583	12179.242167	1
305	28220.000000	53525.200000	92829.400000	3
322	2002.100000	19067.000000	9671.000000	1
342	1640.545455	61886.454545	35012.818182	1
366	491.133333	25522.333333	13672.966667	1
369	28200.000000	43688.000000	115374.000000	3
386	12633.820000	41961.800000	8134.400000	1

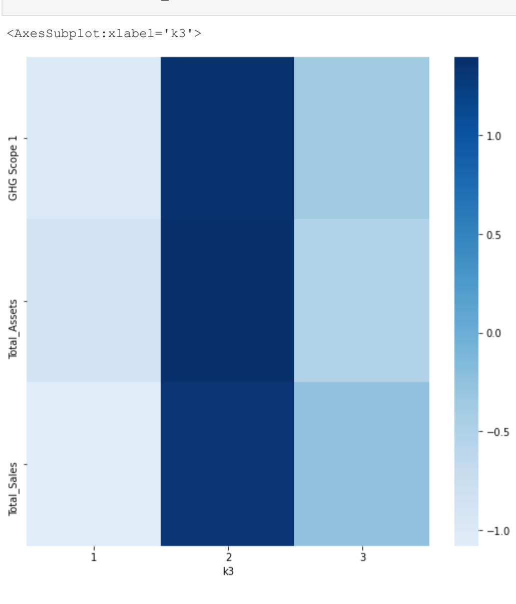
In [78]:
```
clus_profile = stock_numeric.groupby("k3").mean()
clus_profile
```

Out[78]:

k3	GHG Scope 1	Total_Assets	Total_Sales
1	4984.578172	31384.022544	15947.468311
2	96028.571429	262006.000000	252494.392857
3	29586.630357	66245.253571	97467.975000

In [79]:
```
scp = StandardScaler()
cluster_scaled = scp.fit_transform(clus_profile)
cluster_scaled = pd.DataFrame(cluster_scaled, index=clus_profile.index, columns=clus_profile.columns)
plt.figure(figsize=(9,9))
sns.heatmap(cluster_scaled.T, cmap="Blues", center=0)
```

Out[79]: <AxesSubplot:xlabel='k3'>



Start creating classification score within industry

We came up with the following rules to build our classification model.

- Create new column -> GHG Emissions
```
    o Each cluster will represent a value from: Low, Medium, High
```
- Three labels regarding the environmental actions
```
    o Good Forecast
        - In the last 3 years:
            - The % change in GHG scope is descending
            - GHG emissions are below the cluster average
            - The Environmental Disclosure score higher than the average score of the companies in the
cluster
    o Improving
        - In the last 2 years:
            - The % change in GHG scope is descending
            - GHG emissions are below the cluster average
            - The Environmental Disclosure score higher than the average score of the companies in the
cluster
    o Neutral
        Else, then bad.
```

Utility Industry

We will start by assigning the labels to the companies in the utility industry.

In [80]:
```
stocks_utility = pd.read_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Action
stocks_utility.drop(columns='k3', inplace=True)
stocks_utility.head()
```

Out[80]:

	Company	GHG Scope 1	Total_Assets	Total_Sales	Utility	cluster
0	AES CORP (THE)	66456.888889	37643.333333	14251.555556	1	3
1	ALLIANT ENERGY CORP	15811.300000	14436.700000	3427.600000	1	1
2	AMEREN CORP	29028.414286	24878.571429	6063.285714	1	1
3	AMERICAN ELECTRIC POWER CO	96828.785714	64374.571429	16056.042857	1	3
4	AMERICAN WATER WORKS CO INC	60.855556	15455.169667	2980.634000	1	1

In [81]:
```
stocks_utility_num = stocks_utility.select_dtypes('number')
clus_profile = stocks_utility_num.groupby("cluster").mean().reset_index()
clus_profile.sort_values(by='GHG Scope 1')
```

Out[81]:

	cluster	GHG Scope 1	Total_Assets	Total_Sales	Utility
0	1	14072.114399	29086.251791	7404.031962	1
2	3	54375.456363	40817.857028	12508.907826	1
1	2	60309.799460	98939.217071	20328.243838	1

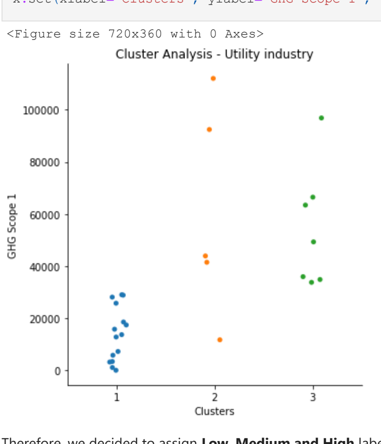
Below, we created a function to assigned a **Low, Medium or High** label to the companies. The conditions were based on if their GHG Scope is equal to minimum, maximum or between the GHG Scope Average.

In [82]:
```
def create_cluster_label(df, field):
    df_grouped = df.groupby(field).mean().reset_index()
    min_ghg = df_grouped['GHG Scope 1'].min()
    max_ghg = df_grouped['GHG Scope 1'].max()

    conditions = [(df_grouped['GHG Scope 1'] == min_ghg),
              (df_grouped['GHG Scope 1'] == max_ghg),
               (df_grouped['GHG Scope 1'].between(left=min_ghg, right=max_ghg))]

    choices = ['Low', 'High','Medium']

    df_grouped['GHG_Emission_category'] = np.select(condlist=conditions, choicelist=choices)
    fields = []
    fields.append(field)
    fields.append('GHG_Emission_category')
    df_grouped = df_grouped[fields]
    df_merged = pd.merge(df, df_grouped, on=field)

    return df_merged
```

However, after analyzing this visualization, we decided that this classification wasn't fair. For example, companies in cluster #2 will be assigned to the **High** label when we can see there are companies that have a similar GHG Scope 1 and are in different clusters

In [83]:
```
plt.figure(figsize=(10,5));
x = sns.catplot(x="cluster", y="GHG Scope 1", data=stocks_utility);
x.set(xlabel='Clusters', ylabel='GHG Scope 1', title = 'Cluster Analysis - Utility industry');
```

<Figure size 720x360 with 0 Axes>



Therefore, we decided to assign **Low, Medium and High** labels within each cluster. In this way, we are being more fair with our classification

In [84]:
```
def splittingIntoGroups(values, num_groups):
    avg = len(values) / float(num_groups)
    out = []
    last = 0.0

    while last < len(values):
        out.append(values[int(last):int(last + avg)])
        last += avg

    np_g = np.array(out)

    return np_g

def assign_labels_by_cluster(df, field):
    total_clusters = len(df[field].unique())
    fields = []
    fields.append('Company')
    fields.append('GHG_Emission_category')
    appended_data = []
    for number in range(total_clusters):
        cluster_num = number+1
        df_stocks = df[df[field] == cluster_num]
        df_values = df[df[field] == cluster_num][['GHG Scope 1']].sort_values(by='GHG Scope 1')['GHG Scope 1'].
        num_companies_inclust = df[df[field] == cluster_num].groupby(field)[['Utility']].count().values[0][0]
        if(num_companies_inclust > 3):
            groups = splittingIntoGroups(df_values, 3)
            if(isinstance(groups[0], list) or isinstance(groups[0], list) or isinstance(groups[0], list)):
                conditions = [(df_stocks['GHG Scope 1'].between(left=np.array(groups[0]).min(), right=np.array
                              (df_stocks['GHG Scope 1'].between(left=np.array(groups[1]).min(), right=np.array
                               (df_stocks['GHG Scope 1'].between(left=np.array(groups[2]).min(), right=np.array
            else:
                conditions = [(df_stocks['GHG Scope 1'].between(left=groups[0].min(), right=groups[0].max())),
                              (df_stocks['GHG Scope 1'].between(left=groups[1].min(), right=groups[1].max())),
                               (df_stocks['GHG Scope 1'].between(left=groups[2].min(), right=groups[2].max()))]
            choices = ['Low', 'Medium','High']
            df_stocks['GHG_Emission_category'] = np.select(condlist=conditions, choicelist=choices)
        else:
            df_stocks['GHG_Emission_category'] = 'Undefined'
        appended_data.append(df_stocks[fields])

    appended_data = pd.concat(appended_data)
    df_merged = pd.merge(df, appended_data, on='Company')
    return df_merged
```

In [85]:
```
cluster_stocks = assign_labels_by_cluster(stocks_utility, 'cluster')
clus_profile1 = cluster_stocks[['cluster','GHG_Emission_category']]
clus_profile1
```

Out[85]:

	cluster	GHG_Emission_category
0	3	High
1	1	Medium
2	1	High
3	3	High
4	1	Low
5	1	Low
6	1	Medium
7	1	Low
8	2	Medium
9	3	Medium
10	2	High
11	1	Low
12	3	Low
13	1	High
14	1	Low
15	2	Low
16	3	Low
17	2	Medium
18	1	High
19	3	High
20	1	Medium
21	1	High
22	1	Medium
23	1	Medium
24	2	High
25	1	High
26	3	Medium

Next, we worked with two functions to implement the rules for the classification labels.

In [86]:
```
def assignEnvironmentalScoreLabel(stocks, companies, cluster_stocks, clus_profile, field):
    company_label_score = {}
    for com in companies:
        data = stocks.loc[stocks.Company == com]
        clust_num = cluster_stocks.loc[cluster_stocks.Company == com,field].tolist() #Get cluster number
        companies_in_cluster = cluster_stocks[(cluster_stocks[field] == clust_num[0]) & (cluster_stocks.Company
        avg_ghg_scope = clus_profile[clus_profile[field] == clust_num[0]]['GHG Scope 1'].tolist() #Get the GHG
        year_max = data.Year.max()
        year_min = year_max - 2

        assigned = checkMetrics(year_max, year_min, data, 'Good Forecast', company_label_score, com, avg_ghg_sc

        if assigned == False:
            year_min = year_max - 1
            data = stocks.loc[stocks.Company == com]
            assigned = checkMetrics(year_max, year_min, data, 'Improving', company_label_score, com, avg_ghg_sc
            if assigned == False:
                company_label_score[com] = 'Neutral'
    return company_label_score
```

In [87]:
```
def checkMetrics(maxYear, min_year, data, label, company_label_score, com, avg_ghg_scope, companies):
    success = False
    com_trend = data.loc[(data.Year <= maxYear) & (data.Year >= min_year), ['Year','Change_in_GHG','Environment
    change_ghg = com_trend['Change_in_GHG'].tolist()
    res = all(abs(i) > abs(j) for i, j in zip(change_ghg, change_ghg[1:])) #Check if the Direct Emissions are d
    if res == True:
        ghg_scope = com_trend['GHG Scope 1'].tolist()
        res_ghg = all(i < avg_ghg_scope[0] for i in ghg_scope) #Check if the company GHG Scope is lower than th
        if res_ghg == True:
            eds_list = com_trend['Environmental Disclosure Score'].tolist()
            eds = stocks.loc[(stocks.Company.isin(companies))&(stocks.Year <= maxYear) &
                             (stocks.Year >= min_year),['Environmental Disclosure Score']].mean().tolist()
            res_eds = all(i < eds[0] for i in eds_list)
            if res_eds == True:
                company_label_score[com] = label
                success = True
    return success
```

In [88]:
```
companies = cluster_stocks.Company.tolist()
my_dict = assignEnvironmentalScoreLabel(stocks, companies, cluster_stocks, clus_profile, 'cluster')
```

In [89]:
```
df_com_utility = pd.DataFrame.from_dict(my_dict, orient='index', columns=['Classification'])
df_com_utility = df_com_utility.reset_index()
df_com_utility.rename(columns = {'index':'Company'}, inplace=True)
df_com_utility.head()
```

Out[89]:

	Company	Classification
0	AES CORP (THE)	Improving
1	ALLIANT ENERGY CORP	Neutral
2	AMEREN CORP	Neutral
3	AMERICAN ELECTRIC POWER CO	Neutral
4	AMERICAN WATER WORKS CO INC	Neutral

In [90]:
```
df_utility = pd.merge(cluster_stocks, df_com_utility, on='Company')
df_utility.head()
```

Out[90]:

	Company	GHG Scope 1	Total_Assets	Total_Sales	Utility	cluster	GHG_Emission_category	Classification
0	AES CORP (THE)	66456.888889	37643.333333	14251.555556	1	3	High	Improving
1	ALLIANT ENERGY CORP	15811.300000	14436.700000	3427.600000	1	1	Medium	Neutral
2	AMEREN CORP	29028.414286	24878.571429	6063.285714	1	1	High	Neutral
3	AMERICAN ELECTRIC POWER CO	96828.785714	64374.571429	16056.042857	1	3	High	Neutral
4	AMERICAN WATER WORKS CO INC	60.855556	15455.169667	2980.634000	1	1	Low	Neutral

Above we can see the companies in the utility industry and their respective Classification. We will proceed to export the dataframe to continue the analysis in another notebook.

In [91]:
```
df_utility.to_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Datasets/
```

Energy Industry

We are going to follow the same 'rules' applied to the utility industry. Also, even thought the clusters have a few observation each, we decided to continue with 3 clusters to be able to assign Low, Medium and High labels.

Eventually, the model could receive more information and the clustering values will get distributed better.

In [92]:
```
stocks_energy = pd.read_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions
stocks_energy.drop(columns='cluster', inplace=True)
stocks_energy.head()
```

Out[92]:

	Company	GHG Scope 1	Total_Assets	Total_Sales	Utility	k3
0	APACHE CORP	14952.566667	36178.852333	10675.480500	0	1
1	BAKER HUGHES CO	373.166667	54286.000000	21324.666667	0	1
2	CABOT OIL & GAS CORP	530.900000	4487.245000	2066.277000	0	1
3	CHEVRON CORP	64592.857143	216105.857143	179877.785714	0	2
4	CONOCOPHILLIPS	37136.521429	121456.214286	101670.000000	0	3

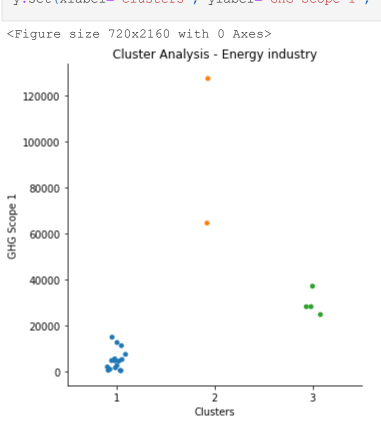
In [93]:
```
stocks_energy_num = stocks_energy.select_dtypes('number')
clus_profile_energy = stocks_energy_num.groupby("k3").mean().reset_index()
clus_profile_energy.sort_values(by='GHG Scope 1')
```

Out[93]:

	k3	GHG Scope 1	Total_Assets	Total_Sales	Utility
0	1	4984.578172	31384.022544	15947.468311	0
2	3	29586.630357	66245.253571	97467.975000	0
1	2	96028.571429	262006.000000	252494.392857	0

In [94]:
```
plt.figure(figsize=(10,30));
y = sns.catplot(x="k3", y="GHG Scope 1", data=stocks_energy);
y.set(xlabel='Clusters', ylabel='GHG Scope 1', title = 'Cluster Analysis - Energy industry');
```

<Figure size 720x2160 with 0 Axes>



In [95]:
```
cluster_stocks = assign_labels_by_cluster(stocks_energy, 'k3')
clus_profile1 = cluster_stocks[['k3','GHG_Emission_category']]
clus_profile1
```

Out[95]:

	k3	GHG_Emission_category
0	1	High
1	1	Low
2	1	Low
3	2	Undefined
4	3	High
5	1	High
6	1	Medium
7	2	Undefined
8	1	Medium
9	1	Medium
10	1	High
11	1	High
12	3	Low
13	1	Low
14	1	High
15	1	Medium
16	3	High
17	1	Medium
18	1	Low
19	1	Low
20	3	Medium
21	1	High

As the table presents, we have some companies with **Undefined**. We will need more than 3 companies in each cluster to be able to compare them and assign a GHG Scope 1 category

In [96]:
```
companies = cluster_stocks.Company.tolist()
dict_energy = assignEnvironmentalScoreLabel(stocks, companies, cluster_stocks, clus_profile_energy, 'k3')
```

In [97]:
```
df_com_energy = pd.DataFrame.from_dict(dict_energy, orient='index', columns=['Classification'])
df_com_energy = df_com_energy.reset_index()
df_com_energy.rename(columns = {'index':'Company'}, inplace=True)
df_com_energy.head()
```

Out[97]:

	Company	Classification
0	APACHE CORP	Neutral
1	BAKER HUGHES CO	Neutral
2	CABOT OIL & GAS CORP	Neutral
3	CHEVRON CORP	Neutral
4	CONOCOPHILLIPS	Neutral

In [98]:
```
df_energy = pd.merge(cluster_stocks, df_com_energy, on='Company')
df_energy.head()
```

Out[98]:

	Company	GHG Scope 1	Total_Assets	Total_Sales	Utility	k3	GHG_Emission_category	Classification
0	APACHE CORP	14952.566667	36178.852333	10675.480500	0	1	High	Neutral
1	BAKER HUGHES CO	373.166667	54286.000000	21324.666667	0	1	Low	Neutral
2	CABOT OIL & GAS CORP	530.900000	4487.245000	2066.277000	0	1	Low	Neutral
3	CHEVRON CORP	64592.857143	216105.857143	179877.785714	0	2	Undefined	Neutral
4	CONOCOPHILLIPS	37136.521429	121456.214286	101670.000000	0	3	High	Neutral

Above we can see the companies in the energy industry and their respective classification. We will proceed to export the dataframe to continue the analysis in another notebook.

In [99]:
```
df_energy.to_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Datasets/E
```

Conclusion

We concluded every step to build our own classification model. As expected with every new model, it is necessary to keep tuning and training this classification so it presents better results. We are confident about the initial steps done to have at least an idea of what is possible. Next steps would be to add more companies for each industry or even keep building new models for other industries.

Moreover, as years pass by and we keep adding more past data, the classification will get updated too and present insightful information to Perpetual Investors.

Next steps

Now, we want to present the classification results in a more friendly and interpretable manner. Therefore, we decided to build a small application using Streamlit to present a dashboard with the results. To see the code related to this application, please refer to the python script **classification_app.py**