Assignment 2-Clustering Techniques in Machine Learning

Siddharth Tripathi
Student Id-19044661
htr0018@autuni.ac.nz

June 2020

Contents

Introduction

In this assignment, we perform Clustering analysis on 4 different types of data sets. Cluster analysis is basically creation of groups or clusters of data in such a way that data in one cluster are more similar to each other compared to data in other cluster. Cluster analysis can be performed by various algorithms. In this assignment, we have performed cluster analysis using 3 algorithms -

1. **K-Means**: K-Means is a centroid based clustering. It works on vector quantization where n observations (or data objects) are clustered into k clusters in such a way that an observation is put into cluster which as nearest mean to that observation.
2. **DBSCAN**: It is a density based clustering algorithm which clusters together closely packed data points and marks the outliers(which have no nearby neighbours) in low density regions.
3. **Agglomerative**: The Agglomerative clustering is basically a hierarchical clustering technique. This type of clustering technique builds a hierarchy of clusters. The Agglomerative clustering has a bottom-up approach. The observation is present in its own cluster. As the algorithm proceeds, the cluster pairs are merged together.

1 Task 1- Implementation of Clustering Techniques - K-MEANS, DBSCAN and AGGLOMERATIVE

In this section we implement the 3 clustering algorithms - K-MEANS, DBSCAN and Agglomerative algorithm on the 4 data sets given in task sheet. We calculate following measure as result of clustering - Sum Squared Error, Cluster Silhouette Measure and Time taken.

Before implementing clustering algorithms on our data sets, we perform data pre-processing. Data pre-processing for each data set were different as per the specific needs of that data set. This includes :

1. Dropping null value columns if all data is null
2. Replacing null values with mean values for a column
3. Dropping columns which were not useful
4. label encoding string data to numeric data
5. Removing special characters and symbols such as $ in price column

Feature Selection Algorithm
For each data set, we implemented Principal Component Analysis(PCA) feature selection algorithm as asked in the assignment task sheet. We reduced dimensionality to 2 components for each data set. We also implemented Standard Scaler for scaling the data for each data set.

```python
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

#Standard Scaler for dow_jones
scale = StandardScaler()
scale.fit(data_jones)
scaled_data = scale.fit_transform(data_jones)


#PCA for dow_jones
pca = PCA(n_components=2)
pca2=PCA(n_components=2)
pca_data_jones = pca.fit_transform(scaled_data)

pca_data_jones2 = pca2.fit_transform(scaled_data)
```

```python
#Pre-processing for sales transaction dataset

#Scaling for sales data
scale = StandardScaler()
scale.fit(data_sales)
scaled_data = scale.fit_transform(data_sales)
#PCA for sales data
pca = PCA(n_components=2)
pca_data_sales = pca.fit_transform(scaled_data)
pca2=PCA(n_components=2)
pca_data_sales2 = pca2.fit_transform(scaled_data)
```

```python
#Pre-processing for water dataset

#Standard Scaler for water dataset
scale = StandardScaler()
scale.fit(data_sales)
scaled_data = scale.fit_transform(data_water)

#PCA for water dataset
pca = PCA(n_components=2)
pca_data_water = pca.fit_transform(scaled_data)
pca2=PCA(n_components=2)
pca_data_water2 = pca2.fit_transform(scaled_data)
```

```python
#Pre-processing for facebook live data set

#Standard Scaler for facebook live dataset
scale = StandardScaler()
scale.fit(data_sales)
scaled_data = scale.fit_transform(data_live)


#PCA for facebook live data set
pca = PCA(n_components=2)
pca_data_live = pca.fit_transform(scaled_data)
pca2=PCA(n_components=2)
pca_data_live2 = pca2.fit_transform(scaled_data)
```

Figure 1: Code implementing PCA for each Dataset

1.1 Task-1 (a) K-Means Algorithm Implementation on 4 Data Sets

TASK - Run the K means algorithm on each of the four datasets. Obtain the best value of K using either SSE and/or CSM. Tabulate your results in a 4 by 3 table, with each row corresponding to a dataset and each column corresponding to one of the three measures mentioned above. Display the CSM plot for the best value of the K parameter for each dataset

In this section, we implement K-Means clustering algorithm on all 4 data sets. As discussed above, these data sets are already pre-processed and feature selection has also been implemented.

4X3 Table for Three Measures - SSE, Silhouette and Time taken

The table(Fig 2) is 4X4 due to one additional measure, Davies Bouldin along with other 3 measures.

Result table

Data Set	Silhoute Score	SSE Score	Davies Score	Time Taken
Dow Jones dataset	0.553142	191.077	0.522847	0.0548179
Facebook Live dataset	0.819889	21512.3	0.596296	0.0568478
Sales and Transactions Dataset	0.667889	17351.6	0.442714	0.0568297
Water Treatment Dataset	0.388325	2736.43	0.767741	0.0308874

Figure 2: 4X4 Table for Measures - K-MEANS

CSM Plot for Best Value of K Parameter for Each Data Set

The best value for K parameter can be observed from CSM plot. We have created two CSM plot - Thickness Plot(Fig 3) and Silhouette Score vs K parameter Graph(Fig 4).

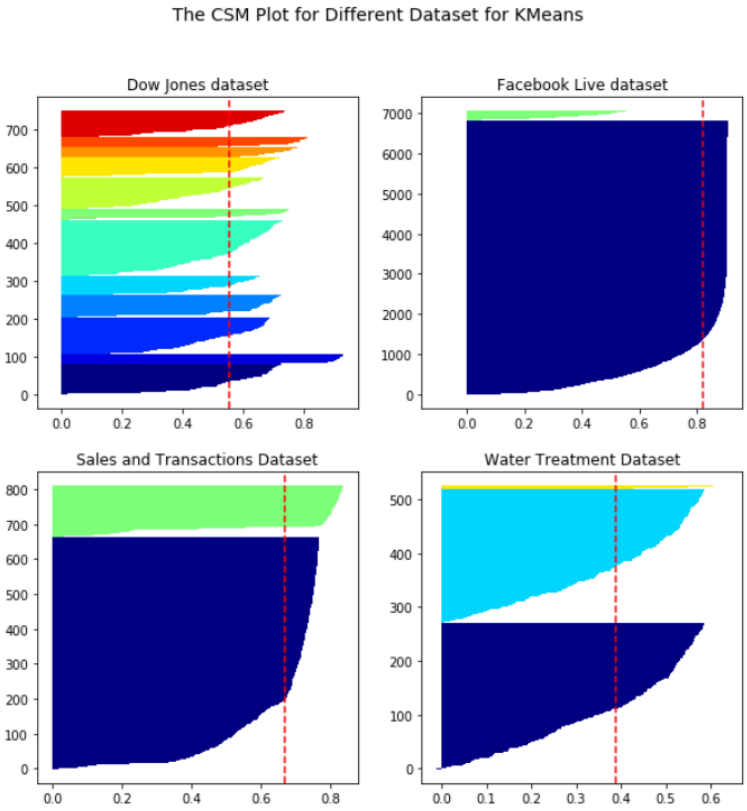

Figure 3: Thickness Curve - CSM (K-MEANS)

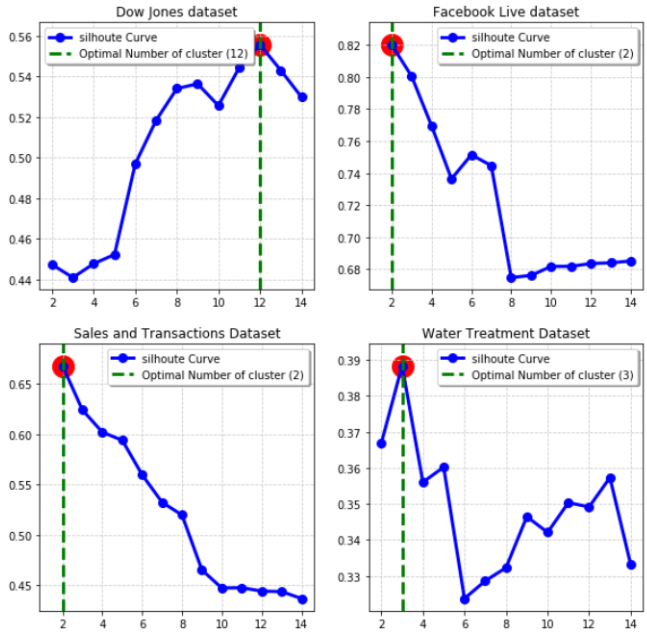

Figure 4: Silhouette Score vs K parameter Graph - CSM (K-MEANS)

1.2 Task-1 (b) DBSCAN Algorithm Implementation on 4 Data Sets

TASK - Repeat the same activity for DBSCAN and tabulate your results once again, just as you did for part a). Display the CSM plot and the 4 by 3 table for each dataset.

In this section, we implement DBSCAN clustering algorithm on all 4 data sets. As discussed above, these data sets are also pre-processed and feature selection has also been implemented to them.

4X3 Table for Three Measures - SSE, Silhouette and Time taken

The table(Fig 5) is 4X4 due to one additional measure, Davies Bouldin along with other 3 measures.

Result table

Data Set	Silhoutte Score	Sum Squared Error	Davies Score	Time Taken
Dow Jones dataset	0.507938	3650.18	0.37703	0.00498748
Facebook Live dataset	0.816399	24514.2	4.27344	0.407367
Sales and Transactions Dataset	0.415743	8783.58	4.31247	0.00397635
Water Treatment Dataset	0.436666	6280.76	3.35608	0.00199175

Figure 5: 4X4 Table for Measures - DBSCAN

CSM Plot for Best Value of K Parameter for Each Data Set

The best value for K parameter can be observed from CSM plot. We have created CSM plot - Thickness Plot(Fig 6)

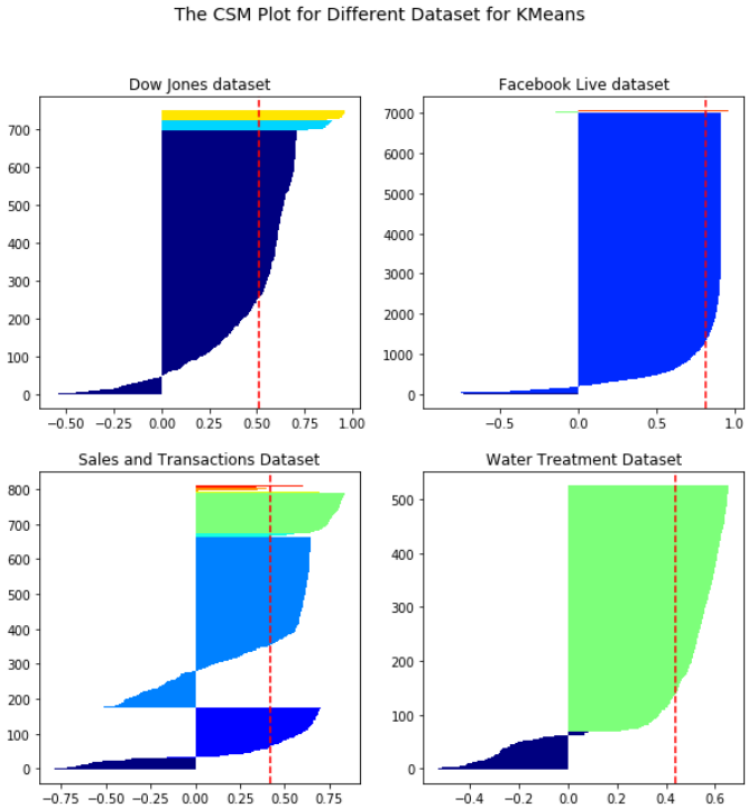

Figure 6: Thickness Curve - CSM (DBSCAN)

1.3 Task-1 (c) Agglomerative Algorithm Implementation on 4 Data Sets

TASK - Finally, use the Agglomerative algorithm and document your results as you did for parts a) and b). Display the CSM plot and the 4 by 3 table for each dataset.

In this section, we implement Agglomerative clustering algorithm on all 4 data sets. Once again, these data sets are already pre-processed and feature selection has already been implemented.

4X3 Table for Three Measures - SSE, Silhouette and Time taken

The table(Fig 7) is 4X4 due to one additional measure, Davies Bouldin along with other 3 measures.

The results for Agglomerative Clustering are as follows:-

Data Set	Silhouette Score	Sum of Square Error	Davies Score	Time Taken
Dow Jones dataset	0.529185	217.156	0.543915	0.0109859
Facebook Live dataset	0.864207	23487.7	0.391833	1.2658
Sales and Transactions Dataset	0.68561	18308.1	0.335554	0.0129478
Water Treatment Dataset	0.385839	2863.67	0.760786	0.00601912

Figure 7: 4X4 Table for Measures - Agglomerative

CSM Plot for Best Value of K Parameter for Each Data Set

The best value for K parameter can be observed from CSM plot. We have created two CSM plot - Thickness Plot(Fig 8) and Silhouette Score vs K parameter Graph(Fig 9) similar to K-Mean.

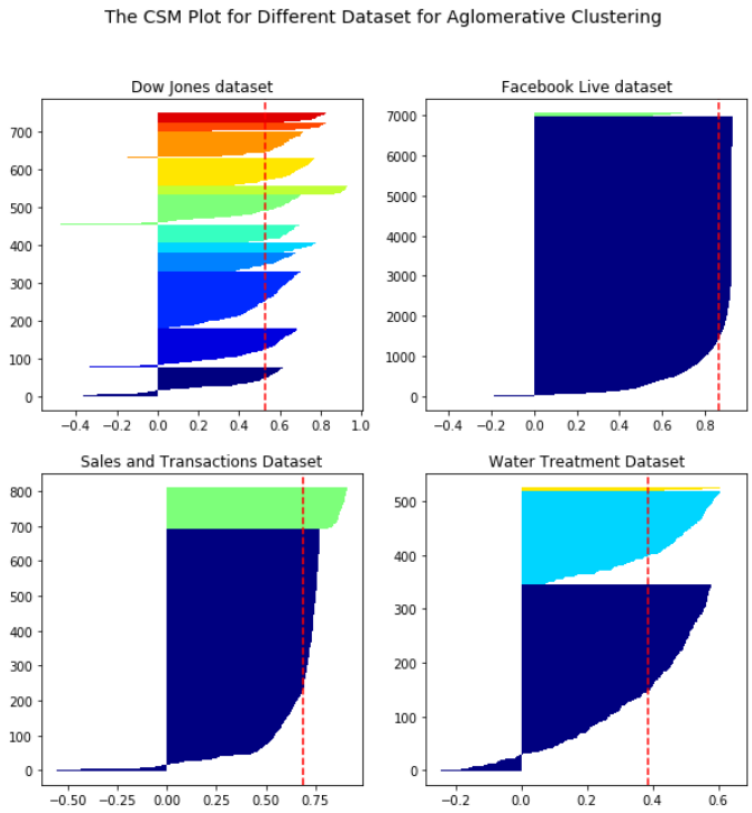

Figure 8: Thickness Curve - CSM (Agglomerative)

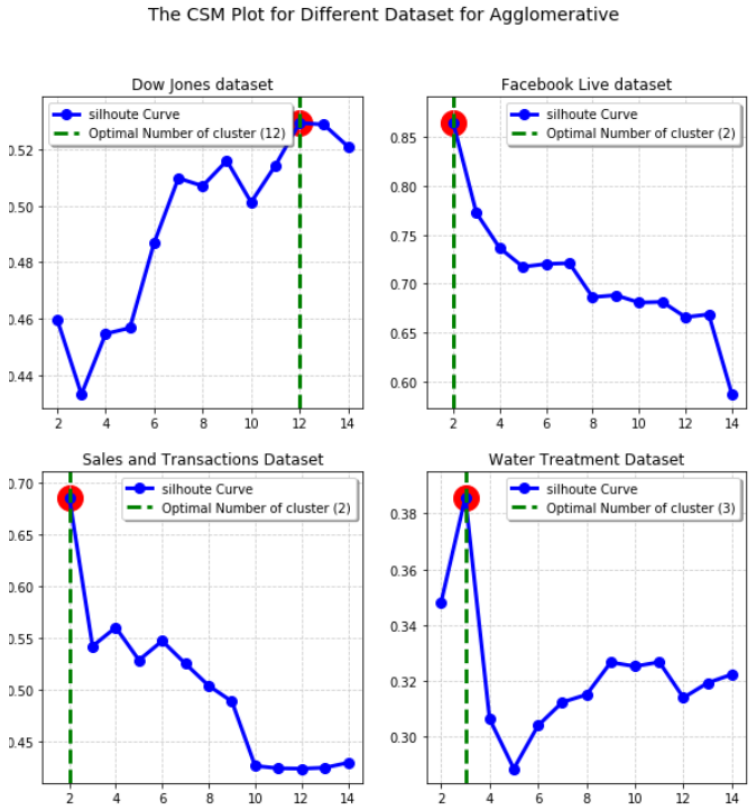

Figure 9: Silhouette Score vs K parameter Graph - CSM (Agglomerative)

1.4 Task-1 Python Code For Data Set DOW JONES FOR PART - A, B, C

TASK - Submit Python code used for parts a) to c) below. You only need to submit the code for one of the 4 datasets.

[30]:
```
import numpy as np
import pandas as pd

from sklearn.datasets import make_blobs
from sklearn.metrics import silhouette_samples
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import matplotlib as mpl
import datetime
import time

import matplotlib.cm as cm

from matplotlib.ticker import FixedLocator, FixedFormatter
from pylab import rcParams
```

[31]:
```
#1 Importing dow data

data_raw = pd.read_csv(r"D:\college\sem2\Data Mining and Machine learning\Assignment␣
↪2\dow_jones_index\dow_jones_index.csv")
```

We perform data pre-processing and data cleaning. Removing null values, dropping columns, unnecessary

[32]:
```
#2 Data Pre-procesing and cleaning

# 2.1-- Checking for null values
data_raw.isnull().sum()
```

[32]:
```
quarter                              0
stock                                0
date                                 0
open                                 0
high                                 0
low                                  0
close                                0
volume                               0
percent_change_price                 0
percent_change_volume_over_last_wk    30
previous_weeks_volume                30
next_weeks_open                      0
next_weeks_close                     0
percent_change_next_weeks_price      0
days_to_next_dividend                0
percent_return_next_dividend         0
dtype: int64
```

[33]:
```
#2.2 Dealing with null values, filling with mean

data_raw.percent_change_volume_over_last_wk = data_raw.
 ↪percent_change_volume_over_last_wk.fillna(data_raw.
 ↪percent_change_volume_over_last_wk.mean())
data_raw.previous_weeks_volume = data_raw.previous_weeks_volume.fillna(data_raw.
 ↪previous_weeks_volume.mean())
data_raw.head()
```

[34]:
```
#2.3 removing the dollar signs

data_raw[data_raw.columns[3:7]] = data_raw[data_raw.columns[3:7]].replace('[\$,]', '',␣
 ↪regex=True).astype(float)
data_raw[data_raw.columns[11:13]] = data_raw[data_raw.columns[11:13]].replace('[\$,]',␣
 ↪'', regex=True).astype(float)
```

[35]:
```
#2.4 Label encoding the non-numeric data into label encode.
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
data_raw['stock']= label_encoder.fit_transform(data_raw['stock'])
print(data_raw['stock'].unique())
```

[36]:
```
#2.5 dropping the data column
data_raw=data_raw.drop(columns='date')
```

[37]:
```
#2.6 Scaling the data. Using StandardScaler

from sklearn.preprocessing import StandardScaler
```

```
scale = StandardScaler()
scale.fit(data_raw)
scaled_data = scale.fit_transform(data_raw)
print(scaled_data)
```

[38]:
```
#2.7 Implementing PCA for feature selection
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

plt.scatter(pca_data[:,0], pca_data[:,1])
```

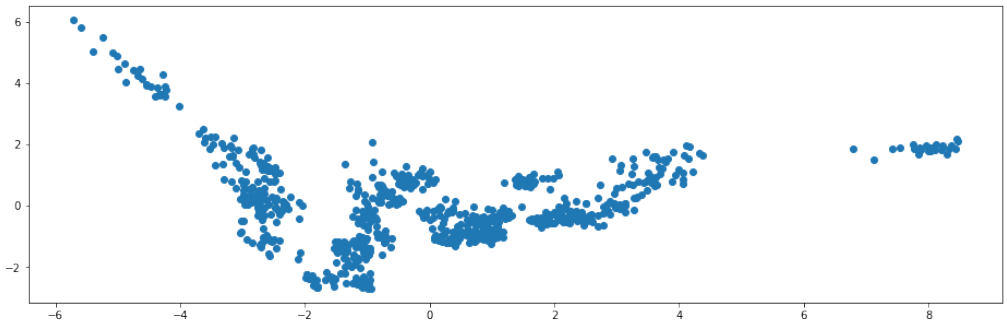

Figure 10: 2-D scatter plot for visualization - PCA

[59]:
```
#Implementing K-Means algorithm for calculating optimal number of clusters by CSM␣
↪analysis.

from sklearn.cluster import KMeans
start_dow = time.time()

k_means_per_k = [KMeans(n_clusters=k,random_state=40).fit(pca_data) for k in␣
↪range(2,15)]

silhoute_scores_per_k = [silhouette_score(pca_data,model.labels_) for model in␣
↪k_means_per_k[0:]]

print(silhoute_scores_per_k)
end_dow = time.time()
time_taken = end_dow-start_dow;
print('TIME TAKEN FOR K-MEANS TO RUN AND CALCULATE SS',time_taken)
```

```
TIME TAKEN FOR K-MEANS TO RUN AND CALCULATE SS 0.6678783893585205
```

[60]:
```
rcParams['figure.figsize']=16,5
```

```
_ = plt.
 ↪plot(range(2,15),silhoute_scores_per_k,'bo-',color='blue',linewidth=3,markersize=8,
label='silhoute Curve')

_ = plt.xlabel('K',fontsize=14,family='Arial')

_ = plt.ylabel('Silhoute Score',fontsize=14,family='Arial')

_ = plt.grid(which='major',color='#cccccc',linestyle='--')

_ = plt.title('Silhoute curve for predicting optimal number of clusters',␣
 ↪family='Arial',fontsize=14)

# #optimal number of k
k  = np.argmax(silhoute_scores_per_k)+2
print('Optimal Number of clusters:-',k)
num_opt_clust = k;
#line to mark optimal number of k in curve
_ = plt.axvline(x=k, linestyle='--', c='green',linewidth=3,label='Optimal Number of␣
 ↪cluster ({})'.format(k))

_ = plt.scatter(k, silhoute_scores_per_k[k-2],c='red',s=400)

_ = plt.legend(shadow=True)

_ = plt.show()
```

```
Optimal Number of clusters:- 12
```

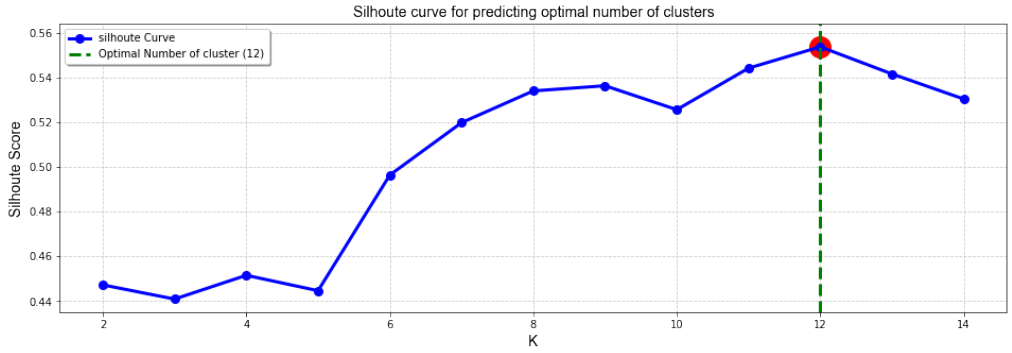

Figure 11: CSM Analysis for optimal value of clusters - K-Means

[61]:
```
#Now from above, best value of K is 9. We can see drop of curve at value 9. Now␣
 ↪running K means for K=9 and plotting CSM
#for the same.
dowjones_data_km = KMeans(n_clusters=9)
start_time = time.time()
dowjones_data_kmfit=dowjones_data_km.fit(pca_data)
dowjones_data_kmeans=dowjones_data_km.fit_predict(pca_data)
```

```
end_time = time.time()
time_run=end_time-start_time
```

[62]:
```
print("Time taken by the KMeans algorithm:-",time_run)
print("The sum of square errors is:-",dowjones_data_kmfit.inertia_)
# data_dow_clustered = pd.concat([data_dow,pd.Series(data_dow_kmeans.labels_)],axis=1)
# data_dow_clustered.rename(columns={data_dow_clustered.columns[-1]:
 ↪'Cluster_number'},inplace=True)
# data_dow_clustered.describe(include='all')
# data_dow_clustered.sort_values(['Cluster_number'])
silhoute_score = silhouette_score(pca_data, dowjones_data_kmfit.labels_)
print("The silhoute score is:-",silhoute_score)
```

```
Time taken by the KMeans algorithm:- 0.09078860282897949
The sum of square errors is:- 299.3384823787691
The silhoute score is:- 0.5362730766435784
```

[65]:
```
#2-D Scatter plot for clusters

data_color=['green','blue','red','yellow','pink','black','grey','orange','violet','purple','Navy','aqua
from matplotlib import cm
for i in range(num_opt_clust):
 plt.scatter(pca_data[dowjones_data_kmeans == i,0],
             pca_data[dowjones_data_kmeans == i,1],
             #s=50,
              c=data_color[i],
             marker='s', edgecolor='black',
             label='cluster '+str(i+1))

plt.legend(scatterpoints=1)
plt.grid()
plt.tight_layout()
#plt.savefig('06_02.png', dpi=300)
plt.show()
```

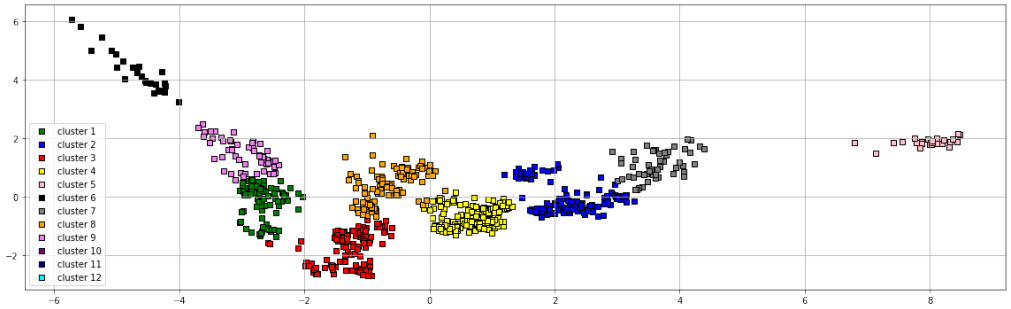

Figure 12: 2-D Scatter plot of clusters - K-Means

[48]:
```
#Making thickness plot based on CSM analysis for dow_jones dataset
```

```
cluster_labels = np.unique(dowjones_data_kmeans)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(pca_data, dowjones_data_kmeans,␣
 ↪metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[dowjones_data_kmeans == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')

plt.tight_layout()
#plt.savefig('06_06.png', dpi=300)
plt.show()
```

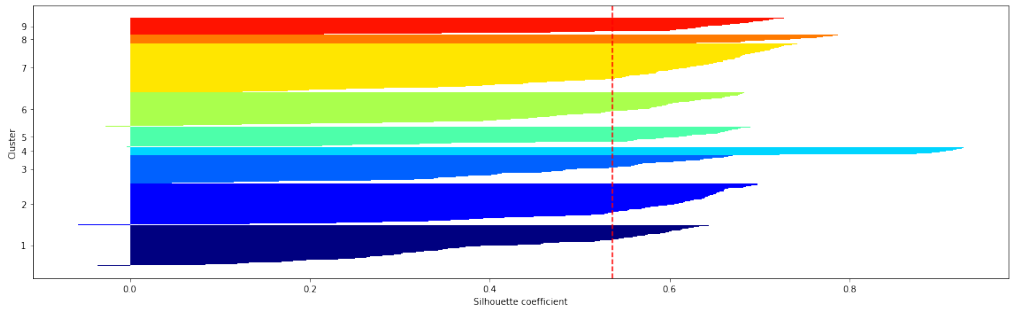

Figure 13: Thickness Curve based on CSM analysis - K-Means

[66]:
```
#Implementing DBSCAN algorithm for calculating optimal eps value for data set Dow_Jones

from sklearn.cluster import DBSCAN
from sklearn import metrics

start= 0.0
stop= 1.0
step = 0.01
my_list = np.arange(start, stop+step, step)
```

```
# print(my_list)
sildbscan=[]
plter=[]
max_eps=0.1
max_silscore=0
for i in range(80):
    db = DBSCAN(eps=.05+my_list[i],min_samples=4)
    data_dbscan = db.fit(pca_data)
    labels = db.labels_
    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
    if(n_clusters_>0):
     sil_score=metrics.silhouette_score(pca_data,labels)
     sildbscan.append(sil_score)
     plter.append(i)
     if(sil_score>max_silscore):
            max_silscore=sil_score
            max_eps=0.05+my_list[i]
# print(sildbscan)
print("Optimal eps for DBSCAN is:",max_eps)
plt.plot(plter,sildbscan)
plt.show()
```

```
Optimal eps for DBSCAN is: 0.77
Optimal K Parameters is: 4
```

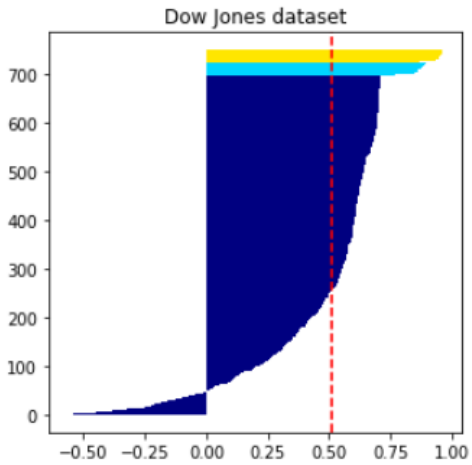

Figure 14: CSM Thickness Curve for optimal K value - DBSCAN

[71]:
```
#Now using DBSCAN algorithm and 2-D scatter plot for cluster visualization

from sklearn.cluster import DBSCAN
from sklearn import metrics
db=DBSCAN(eps=.41, min_samples=4)
data_dbscan=db.fit(pca_data)
data_dbs=db.fit_predict(pca_data)
```

```
# core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
# core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
# print(labels)
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)
print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
print("Silhouette Coefficient: %0.3f"% metrics.silhouette_score(pca_data, labels))
data_color=['red','green','blue','yellow']
for i in range(-1,n_clusters_):
 plt.scatter(pca_data[data_dbs == i,0],
            pca_data[data_dbs == i,1],
            s=50, c=data_color[i],
            marker='s', edgecolor='black',
            label='cluster '+str(i+1))

plt.legend(scatterpoints=1)
plt.grid()
plt.tight_layout()
#plt.savefig('06_02.png', dpi=300)
plt.show()
```

```
Estimated number of clusters: 4
Estimated number of noise points: 9
Silhouette Coefficient: 0.391
```

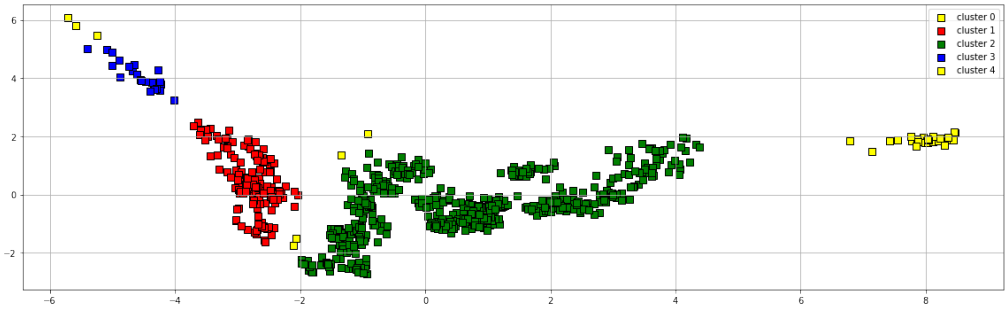

Figure 15: 2-D Scatter Plot visualizing clusters and Noise-DBSCAN

[72]:
```
cluster_labels = np.unique(data_dbs)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(pca_data, data_dbs, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[data_dbs == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
```

```
#     if(c_silhouette_vals[i]>=0):
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)
    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)



silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')

plt.tight_layout()
#plt.savefig('06_06.png', dpi=300)
plt.show()
```

Agglomerative Clustering Implementatio

[164]:
```
#Implementing agglomerative clustering for calculating optimal number of clusters.

from sklearn.cluster import AgglomerativeClustering

agglomerative_per_k = [AgglomerativeClustering(n_clusters=k).fit(pca_data) for k in␣
 ↪range(2,15)]

#print(agglomerative_per_k)

silhoute_scores_per_k = [silhouette_score(pca_data,model.labels_) for model in␣
 ↪agglomerative_per_k[0:]]

print(silhoute_scores_per_k)
```

[73]:
```
rcParams['figure.figsize']=16,5

_ = plt.
 ↪plot(range(2,15),silhoute_scores_per_k,'bo-',color='blue',linewidth=3,markersize=8,
label='silhoute Curve')

_ = plt.xlabel('K',fontsize=14,family='Arial')

_ = plt.ylabel('Silhoute Score',fontsize=14,family='Arial')

_ = plt.grid(which='major',color='#cccccc',linestyle='--')

_ = plt.title('Silhoute curve for predicting optimal number of clusters',␣
 ↪family='Arial',fontsize=14)

# #optimal number of k
k  = np.argmax(silhoute_scores_per_k)+2
print('Optimal Number of Clusters:-',k)
```

```
num_opt_clust = k;
#line to mark optimal number of k in curve
_ = plt.axvline(x=k, linestyle='--', c='green',linewidth=3,label='Optimal Number of␣
 ↪cluster ({})'.format(k))

_ = plt.scatter(k, silhoute_scores_per_k[k-2],c='red',s=400)

_ = plt.legend(shadow=True)

_ = plt.show()
```

```
Optimal Number of Clusters:- 12
```

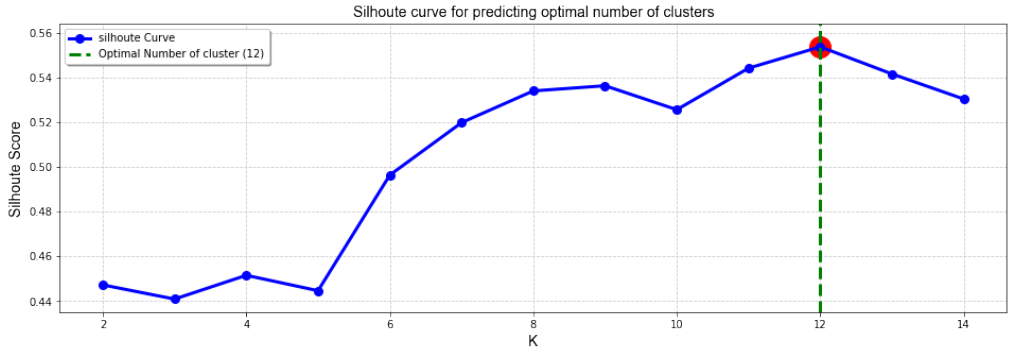

Figure 16: CSM Curve using Aglomerative algorithm for Optimal Number of Clusters

[74]:
```
#2-D scatter plot for data Visualization of clusters

data_color=['green','blue','red','yellow','pink','black','grey','orange','violet','purple','Navy','aqua
from matplotlib import cm
for i in range(num_opt_clust):
 plt.scatter(pca_data[dowjones_data_kmeans == i,0],
            pca_data[dowjones_data_kmeans == i,1],
            #s=50,
             c=data_color[i],
            marker='s', edgecolor='black',
            label='cluster '+str(i+1))

plt.legend(scatterpoints=1)
plt.grid()
plt.tight_layout()
#plt.savefig('06_02.png', dpi=300)
plt.show()
```

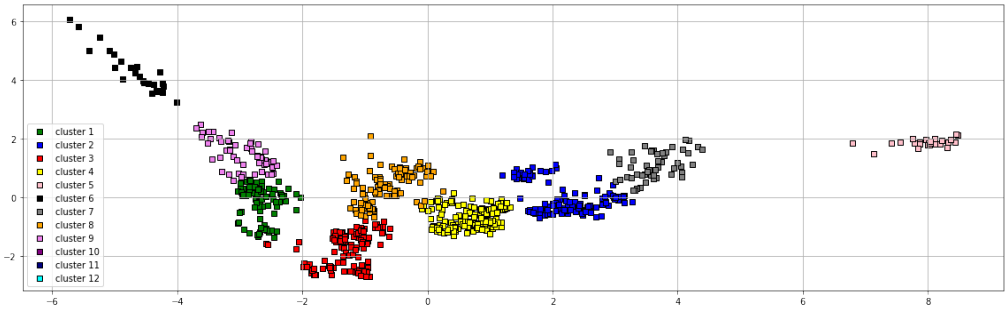

Figure 17: 2-D Scatter Plot for Data Visualization

[ ]:
```
#Making thickness plot based on CSM analysis for dow_jones dataset

cluster_labels = np.unique(dowjones_data_Agglo)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(pca_data, dowjones_data_Agglo, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[dowjones_data_Agglo == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')

plt.tight_layout()
#plt.savefig('06_06.png', dpi=300)
plt.show()
```

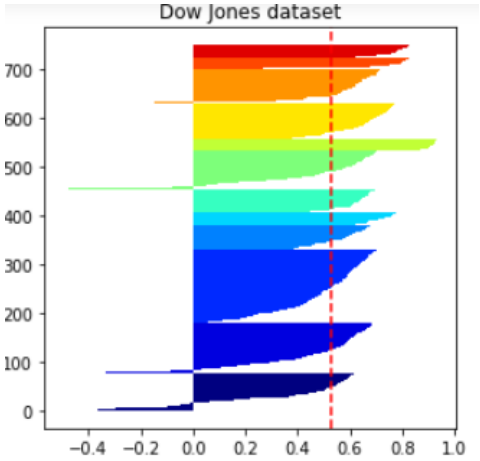

Figure 18: CSM Thickness Curve - DOW JONES (Agglomerative)

2 Task 2 - Performance Measure for Clustering Algorithm

In this section, we evaluate the best performing clustering algorithm out of the 3 algorithms, for each data set and best clustering algorithm overall.

2.1 Task 2(a) - Best Performing Algorithm For Each Data Set

For each dataset identify which clustering algorithm performed best. Justify your answer.In the event that no single algorithm performs best on all three performance measures you will need to carefully consider how you will rate each of the measures and then decide how you will produce an overall measure that will enable you to rank the algorithms.

The clustering analysis is done for 4 different data sets as provided in this assignment. For each data set, we calculate a set of different performance measures which are as follows:

1. Silhouette Score
2. Sum Squared Error
3. Davies Bouldin
4. Time Taken By each Algorithm for Each Data Set

Based on above measures, we can do performance analysis of winner clustering algorithm for each data set and overall winner. In addition to that, we have also done Cluster Silhouette Measure Analysis to calculate optimal number of clusters and eps value(in case of DBSCAN), for each data set.

For each data set, we perform 3 common data pre-processing steps, before we implement clustering algorithms on them. These are as follows:

1. Data Cleaning - Handling null values, handling discontinuous data, dropping column, label encoding etc.
2. Data Scaling using Standard Scaler
3. Dimensionality reduction using PCA algorithm. The dimensionality is reduced to 2 components for each data set as it gives best performance measures and helps in clear performance analysis.

Now lets do performance analysis. For better and clear understanding, a combined table of all the measures for all data sets from all clustering algorithm (Fig 19).

Performance Measure from K-Means clustering for each Dataset

Data Set	Silhouette Score	Sum of Square Error	Davies Score	Time Taken
Dow Jones dataset	0.553142	191.077	0.522847	0.0548179
Facebook Live dataset	0.819889	21512.3	0.596296	0.0568478
Sales and Transactions Dataset	0.667889	17351.6	0.442714	0.0568297
Water Treatment Dataset	0.388325	2736.43	0.767741	0.0308874

Performance Measure from DBSCAN clustering for each Dataset

Data Set	Silhouette Score	Sum of Square Error	Davies Score	Time Taken
Dow Jones dataset	0.507938	3650.18	0.37703	0.00498748
Facebook Live dataset	0.816399	24514.2	4.27344	0.407367
Sales and Transactions Dataset	0.415743	8783.58	4.31247	0.00397635
Water Treatment Dataset	0.436666	6280.76	3.35608	0.00199175

Performace Measure from Agglomerative clustering for each Dataset

Data Set	Silhouette Score	Sum of Square Error	Davies Score	Time Taken
Dow Jones dataset	0.529185	217.156	0.543915	0.0109859
Facebook Live dataset	0.864207	23487.7	0.391833	1.2658
Sales and Transactions Dataset	0.68561	18308.1	0.335554	0.0129478
Water Treatment Dataset	0.385839	2863.67	0.760786	0.00601912

Figure 19: Performance Measure Table for each data set and clustering algorithm

2.1.1 How Do We Decide Winner Algorithm

There can be many different ways through which cluster quality can be measured. In task 1, we calculate 4 measures for each data set using three different clustering techniques.

1. Silhouette Score: The Silhouette Score is a measure which describes how well a data point fits in its respective cluster its distance with centroid. Also Silhouette Score is considered as an optimal performance measure for clustering techniques and is implacable on most clustering techniques. It is computationally more demanding but it is more informative compared to Davies Bouldin and Sum Square Error.
2. Sum Squared Error: This measures the distance between data sample and centroid of cluster for each sample present in that cluster. However, SSE is not optimal for data which has density based clusters or non circular clusters. Thus for algorithms like DBSCAN and Agglomerative it is not an optimal measure. Hence this can be ruled out while deciding winner.
3. Time-Taken: This factor is not a factor which impacts clustering quality. The time taken may fluctuate depending upon processor speed. Also, with respect to our assignment, there is no significant difference in time taken by algorithms for each data set. Hence this measure has a low influnce in deciding performance measure.
4. Davis Bouldin: This measure considers the ratio of cluster scatter and cluster separation based on which similarity between clusters is calculated. If Davies Bouldin value is low, it implies that clustering is better.

Different Measures can be used as per the requirement. As per requirement of this assignment, for our performance analysis, we have taken Silhouette Score measure as a deciding measure for winner algorithm for each data set due to following reason:

- Silhouette measure is more informative compared to other measures
- Results are more accurate compared to other measures
- Applicable on all 3 clustering algorithms we used unlike Sum Squared Error which is not optimal for DBSCAN and Agglomerative

The Silhouette Score based performance analysis for winner algorithm is as follows

2.1.2 DOW-JONES DATASET - WINNER K-MEANS

For the Dow Jones data set, we obtain 3 different Silhouette Scores from 3 clustering algorithms which are as follows:

1. K-Means - 0.55
2. DBSCAN - 0.51
3. Agglomerative - 0.52

RESULT: The best Silhouette score is obtained from K-Means algorithm followed by agglomerative and DBSCAN respectively.

2.1.3 FACEBOOK LIVE DATASET - WINNER AGGLOMERATIVE

For the Facebook Live data set, we obtain 3 different Silhouette Scores from 3 clustering algorithms which are as follows:

1. K-Means - 0.81
2. DBSCAN - 0.81
3. Agglomerative - 0.86

RESULT: The best Silhouette score is obtained from Agglomerative algorithm followed by DBSCAN and K-Means which have same SS.

2.1.4 SALES AND TRANSACTION DATASET - WINNER AGGLOMERATIVE

For the Facebook Live data set, we obtain 3 different Silhouette Scores from 3 clustering algorithms which are as follows:

1. K-Means - 0.66
2. DBSCAN - 0.41
3. Agglomerative - 0.69

RESULT: The best Silhouette score is obtained from Agglomerative algorithm followed by DBSCAN and K-Means which have same SS.

2.1.5 WATER TREATMENT DATASET - WINNER DBSCAN

For the Facebook Live data set, we obtain 3 different Silhouette Scores from 3 clustering algorithms which are as follows:

1. K-Means - 0.39
2. DBSCAN - 0.41
3. Agglomerative - 0.38

RESULT: The best Silhouette score is obtained from DBSCAN algorithm followed by DBSCAN and K-Means which have same SS.

2.2 Task 2(b) - Why Winner Algorithms Produced Best Value for CSM

For each winner algorithm and for each dataset explain why it produced the best value for the CSM measure. This explanation must refer directly to the conceptual design details of the algorithm. There is no need to produce any further experimental evidence for this part of the question.

As we saw in previous answer, the winner algorithm in each data set is producing the best CSM measure. An obvious question which arises is why is it so? In this section we answer this question, by analysing 2-D Scatter plot visualization of each data set and insights provided through it.

2.2.1 DOW JONES DATASET

The winner algorithm for this data set is K-Means. The Silhouette Score(0.55) for this dataset was highest for K-Means algorithm among the three Clustering techniques. K-Mean Algorithm produces best results when the data set is distinct as well as separated in linear manner. When cluster center number is clearly specified because of well defined and separated data(Nathan Landman, 2017).

In the following visualization(Fig 20) of our respective data set, we see that clusters can be clearly differentiated from one another(boxes and circles highlighting the distinct clusters formed. Based on this insight, K-Means clustering algorithm is the right choice for this data set. And that comes out to be correct as K-Means algorithm produces highest CSM measure for this data set.

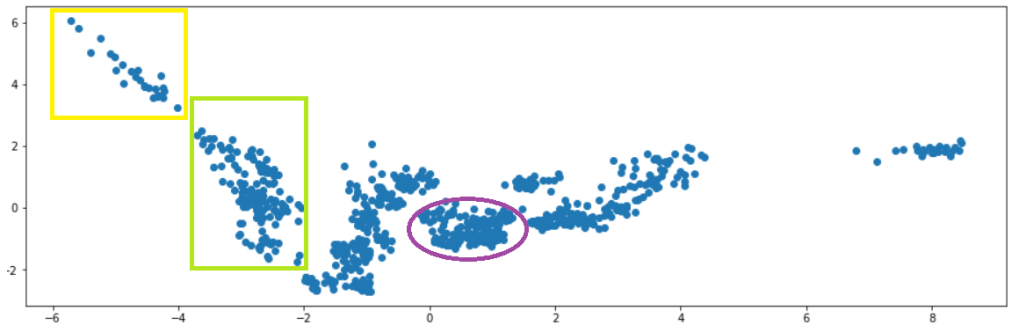

Figure 20: Visualization - DOW JONES 2-D Scatter

2.2.2 FACEBOOK LIVE DATASET

The winner algorithm for this data set is Agglomerative Algorithm. The Silhouette Score(0.86) for this data set was highest for Agglomerative among the three Clustering techniques. Agglomerative Clustering is a hierarchical clustering which is also known as connectivity based clustering. The main idea behind this type of clustering is that clusters are formed for objects similar to nearby objects rather than objects farther away. Agglomerative means aggregating objects into clusters in a hierarchical manner, starting with single object like a Tree structure(Kilitcioglu, 2018).
Distance within the cluster is selected. Each data object is made as separate cluster. Two clusters having minimum inter-cluster distance are merged until a satisfactory clustering is obtained(Forsyth, 2019).

In the following visualization(fig 21) of the facebook live data set, we see that starting from single element as clusters, merging of clusters is done unless a satisfactory cluster is formed. The data set is scattered and merging into clusters later. In the 2-D visualization, we can see cluster formation in hierarchical form(arrows). From Arrow 1, the data points which were scattered are getting merged resulting in cluster formation towards arrow 3

Hence, looking at this visualization, Agglomerative Clustering is most optimal technique which is prooved by the high CSM score for this data set when Agglomerative is applied.

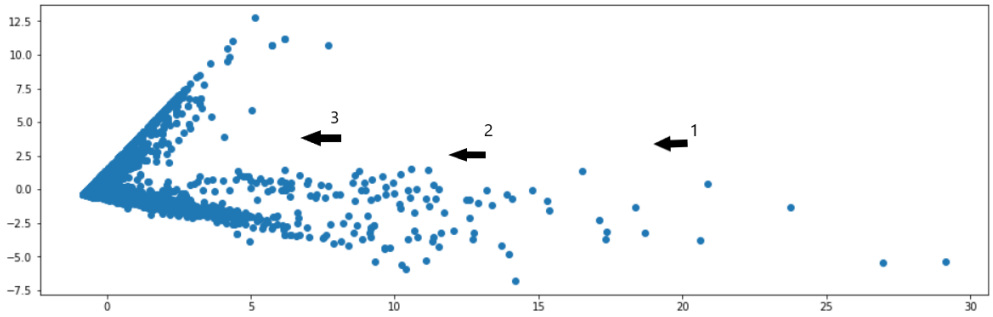

Figure 21: Visualization - Facebook Live 2-D Scatter

2.2.3 SALES AND TRANSACTION DATASET

The winner algorithm for this data set is Agglomerative Algorithm. The Silhouette Score(0.69) for this data set was highest for Agglomerative among the three Clustering techniques. We already discussed about agglomerative clustering in Facebook Live Data Set section.

In the following visualization(fig 22) of the Sales and Transaction data set, we see that starting from single element as clusters, merging of clusters is done unless a satisfactory cluster is formed. The data set is scattered and merging into clusters later. In the 2-D visualization, we can see cluster formation in hierarchical form(arrows). From Arrow 3, the data points which were scattered are getting merged resulting in cluster formation towards arrow 1

Hence, looking at this visualization, Agglomerative Clustering is most optimal technique which is prooved by the high CSM score for this data set when Agglomerative is applied.

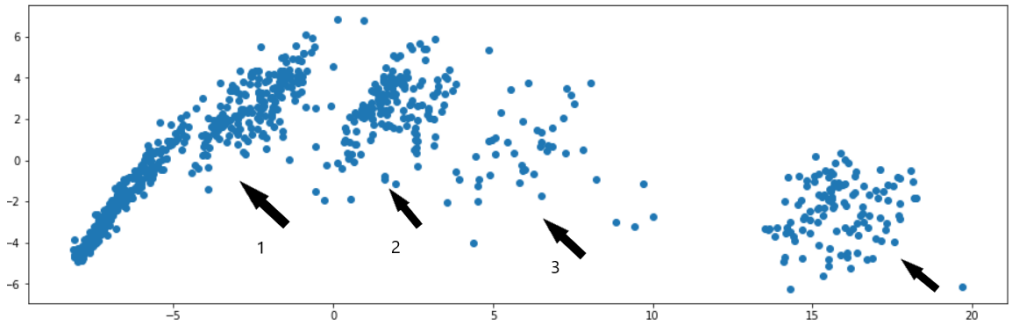

Figure 22: Visualization - Sales And Transaction 2-D Scatter

2.2.4 WATER TREATMENT DATASET

The winner algorithm for this data set is DBSCAN Algorithm. The Silhouette Score(0.44) for this data set was highest for DBSCAN among the three Clustering techniques. DBSCAN is a density based clustering. In this clustering, areas of higher density in visualization are defined as clusters where as the sparse points which are away are considered as noise or cluster boundary. One advantage of density based clustering (DBSCAN) is that it does not require optimal cluster value as input(campello2020density). For proper detection of clusters and its borders, the algorithm expects a density drop in data. This is often considered as a drawback since it produces arbitrary results for overlapping data.

In the following visualization(fig 23) of the Sales and Transaction data set, we see that the data set is denser in the center and as we move away from center, the density reduces and data objects get scattered (outliers are also marked in DBSCAN in large number).The Cluster formation is density based for this data set and

hence we can say that density based clustering algorithms (DBSCAN in our case) will be most suitable for this data set. This assumption is right as for this data set DBSCAN is giving highest Silhouette Score.

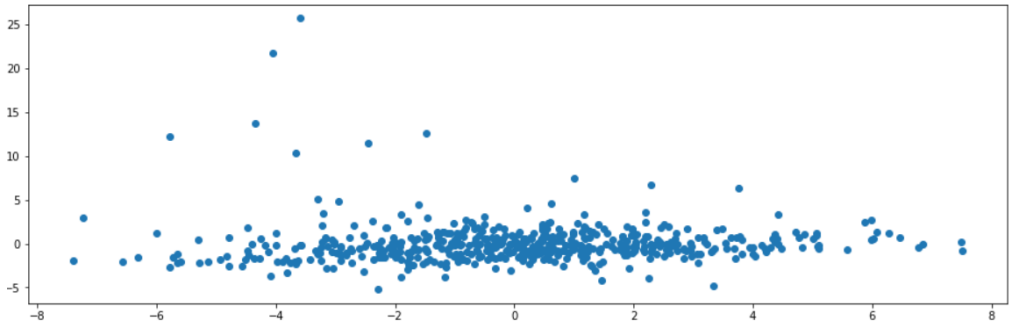

Figure 23: Visualization - Sales And Transaction 2-D Scatter

2.3 Task 2(c) - Best Overall clustering Algorithm

TASK - Based on what you produced in a) above, which clustering algorithm would you consider to be the overall winner (i.e. after taking into consideration performance across all four datasets). Justify your answer.

In the previous sections, we performed a performance analysis based on Cluster Silhouette Measure as it was an optimal and applicable measure on all 3 clustering algorithms. Based on it, we chose a winner algorithm in case of each data set.

In this section, we calculate overall winner algorithm based on its average performance measure for all data sets as in following table(Fig 24):

Data Set	Algorithm	Silhouette Score	Sum of Square Error	Davies Score	Time Taken
Dow Jones	K-Means	0.553142	191.077	0.522847	0.0548179
Facebook Live	K-Means	0.819889	21512.3	0.596296	0.0568478
Sales and Transction	K-Means	0.667889	17351.6	0.442714	0.0568297
Water Treatment	K-Means	0.388325	2736.43	0.767741	0.0308874
Average		**0.60731125**	**10447.85175**	**0.5823995**	**0.0498457**
Dow Jones	DBSCAN	0.507938	3650.18	0.37703	0.00498748
Facebook Live	DBSCAN	0.816399	24514.2	4.27344	0.407367
Sales and Transction	DBSCAN	0.415743	8783.58	4.31247	0.00397635
Water Treatment	DBSCAN	0.436666	6280.76	3.35608	0.00199175
Average		**0.5441865**	**10807.18**	**3.079755**	**0.104580645**
Dow Jones	Agglomerative	0.529185	217.156	0.543915	0.0109859
Facebook Live	Agglomerative	0.864207	23487.7	0.391833	1.2658
Sales and Transction	Agglomerative	0.68561	18308.1	0.335554	0.0129478
Water Treatment	Agglomerative	0.385839	2863.67	0.760786	0.00601912
Average		**0.61621025**	**11219.1565**	**0.508022**	**0.323938205**

Figure 24: Overall Winner Table

Cluster Silhouette Measure (Silhouette Score

The best clustering algorithm overall across all data sets in above analysis is Agglomerative Clustering. This is clearly visible from the average Cluster Silhouette Measure for the algorithm(highlighted green). The runner-up algorithm is K-means while DBSCAN is last(marked in orange). We already discussed in part a of TASK 2 that why we took Silhouette measure as deciding measure of winning algorithm. Also, Silhouette Score gives slightly more accurate results compared to Davies Bouldin Index(Petrovic, 2006).

Davies Bouldin Index

The best clustering algorithm overall across all data sets for this measure is also Agglomerative Clustering. As we discussed earlier, lower Davies Bouldin Index means better clustering. In our case, Agglomerative clustering has lowest DBI which is followed by K-MEANS. DBSCAN is very poor Davies Bouldin Index (marked with orange)

RESULT

From above discussion, we can say that Agglomerative Algorithm performs best on two measures CSM(our decisive measure) and Davis Bouldin Index. These two measures are most important measure for cluster quality.

K-Means takes lesser time compared to other two algorithms which conclude that it is the least computationally demanding among three clustering algorithms.

3 Task 3 - t-distributed Stochastic Neighbor Embedding(t-SNE)

In this task, we research about t-SNE technique which is used for data visualization of high-dimensionality data. We implement the technique on of the data set we used previously (Sales_data).

t-SNE is a non-linear technique for data visualization and exploration. It is an unsupervised technique used for high dimensional data. Since its development in 2008, it has been one of the, most used, accurate and understandable technique for data exploration. It can visualize clusters of data points which are of high dimension, even with very little tuning of parameters(Pezzotti et al., 2019).

The one obvious question which comes up is - How t-SNE is different from PCA?
First significant difference is that PCA is effective on linear data as it is a linear dimensionality reduction technique. This often leads to poor visualization of data, specially for non-linear and high dimensional data, as things that are different end up far away because PCA tends to preserve large pairwise distances(Global Similarities). On the other hand, t-SNE is opposite to PCA when it comes to pairwise distance. t-SNE preserves only small distances(known as local similarities). This helps in better data visualization for high dimensional and non-linear data sets.

Another difference is that PCA was developed way before t-SNE. Unlike t-SNE which was proposed very recently(2008), PCA was proposed way back in 1933. Since 1933 their have been significant changes in computation techniques, computational devices, size of data etc.

3.1 Task 3(a)- Potential Advantage of t-SNE over PCA

TASK - After gaining an understanding of how it works identify one important potential advantage of t-sne over Principal Components Analysis (PCA). You may use one or more sources from the machine learning literature to support your answer.

One potential and most significant advantage of t-SNE over PCA is better data visualization of high dimensional data compared to PCA. This better visualization has following factors, which are absent in PCA, making t-SNE a better choice:

1. **Crowding problem**:The area to accommodate smaller pair-wise distance is more compared to area available for accommodating pair-wise distance which are moderate. In order to map the close points

in proper manner, moderate distant points are pushed far away. The result is that moderate distances are squashed in lower dimension, removing the gaps between original clusters and it appears like a one big cluster(Maaten and Hinton, 2008). t-SNE overcomes this crowding issue by using Student's t-distribution instead of Gaussian Distribution for describing similarity in data in lower dimension(the one which is congested) (Song et al., 2019).

2. **Outliers**:It has been noticed that outliers have significant impact on performance of PCA. One of the reason for that is that PCA does not capture local data characterstics very well. Therefore, PCA is applied after removal of outliers. t-SNE on other side, captures precise local data structures, preserves global structures and shows both outliers and main clusters(Li et al., 2017).

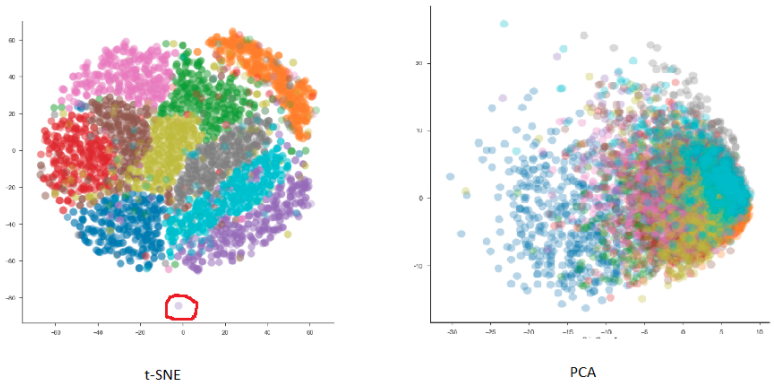

Figure 25: Visualization - t-SNE vs PCA(Violante, 2018)

In fig 25, we can see that in t-SNE visualization, the main clusters and outliers(marked in red circle) are shown simultaneously. Crowding issue is also not there in this visualization. Whereas in PCA, outliers are not properly shown and their is crowding issue as well. Main clusters are not visualized properly.

3.2 Task 3(b)- Implementing t-SNE on Sales data

TASK - Apply t-sne to reduce dimensionality to two components and then visualize the data using a suitable plot. Submit the Python code and your plot

In this section, implementation of the t-SNE algorithm has been done on the Sales-Transaction data set.

3.2.1 Task 3(b)-1: Implementing t-SNE using Python Code and Plot

[7]:
```
from sklearn.manifold import TSNE
import numpy as np
import pandas as pd
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib.ticker import FixedLocator, FixedFormatter
```

[8]:
```
#Importing Data Set
data_sales = pd.read_csv(r"D:\college\sem2\Data Mining and Machine learning\Assignment␣
↪2\Sales_Transactions_Dataset_Weekly.csv");
data_sales.head()
```

[9]:
```
#Data Cleaning as in task 1
data_sales=data_sales.drop(columns='Product_Code')
```

Below cell implements t-SNE dimensionality reduction technique. The dimensionality is reduced to 2 by giving value 2 to parameter n_component

[14]:
```
#Implementing t-SNE dimensionality reduction

tsne = TSNE(n_components=2)
tsne_data_sales = tsne.fit_transform(data_sales)

plt.scatter(tsne_data_sales[:,0], tsne_data_sales[:,1])
```

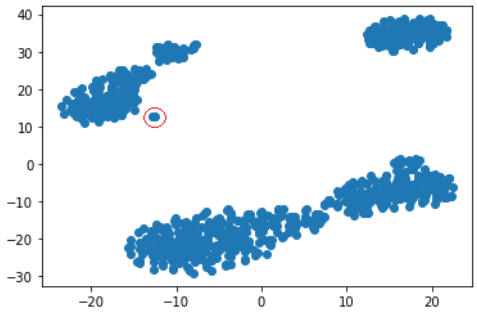

Figure 26: 2-D visualization of Sales-Transaction data set

3.2.2 Task 3(b)-2: Presence of Potential Advantage In Data Set

TASK - Is the potential advantage of t-sne over PCA that you mentioned in part a) present itself in this dataset? Justify your answer with suitable experimental evidence

In part a of task 3, we identified a potential advantage of t-SNE over PCA, that was better visualization. In this section, we implement both the algorithms on our data set Sales-Transaction and generate a plot for 2-D visualization(Fig 27).

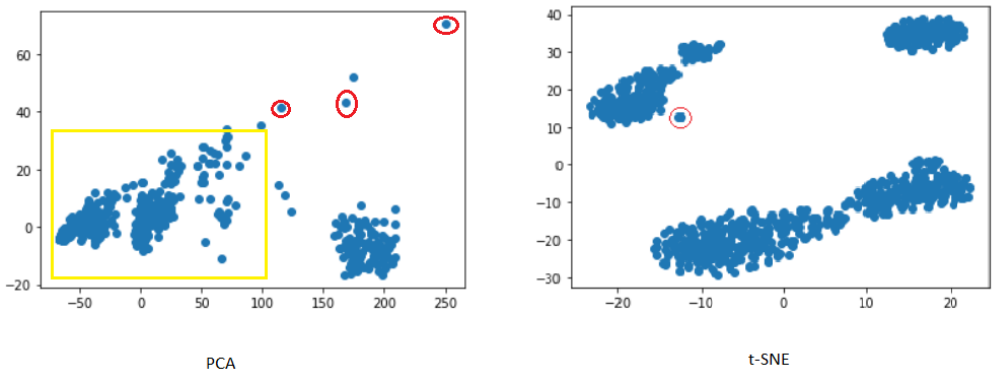

Figure 27: 2-D visualization of Sales-Transaction data - PCA vs t-SNE

The above plots are experimental evidence, from which we can deduce some key points of comparison for both PCA and t-SNE:

1. The t-SNE gives clear insights into structure of data , unlike the PCA which does not give a clear visualization.

2. The t-SNE provides better cluster definition while PCA shows fragmentation of clusters. In addition, t-SNE plot also shows the shape of clusters
3. In PCA plot, due to squashing in lower dimensions, we see crowding issue (marked in yellow box) making it appear like one big cluster. Also, the outliers (marked in red) are not properly separate from clusters and hence produce an unclear visualization. On the other hand, t-SNE plot is more clear. Clusters as well as outliers(marked in red) are well separated and visualized in a more proper and clear way.Hence we can say that the potential advantage we identified for t-SNE in part a exists for our data set- Sales Transaction.

Hence we can say that the potential advantage(Better Visualization) we identified for t-SNE in part a exists for our data set- Sales Transaction and it can be said that t-SNE is better than PCA

3.2.3 Task 3(b)-3: Insights into structures of Data

TASK - Does the 2D visualization give insights into the structure of the data? Explain your answer

The data visualization of data set gives various insights about data set. Insights are indicators of structures in a data. Insights can help us select suitable clustering algorithm for our data set. The 2-D visualization for data set sales transaction using t-SNE (Fig 26) gives us following insight about data:

1. The well defined clusters help us select the suitable clustering algorithm for our data set.
2. Outliers of data set are marked distinctly. This helps us to know number of outliers and whether the data requires pre-processing in case outliers are high.
3. Clearly separated clusters also help us to know the optimal number of clusters for data set.

3.2.4 Task 3(b)-4: Choice of Clustering Algorithm

TASK - If so, does it inform the choice of which clustering algorithm to use? On the other hand, if it does not narrow down the choice of clustering algorithm then explain why the visual is insufficient on its own draw a definite conclusion.

The insights which we observed from the data visualization by t-SNE, the clustering algorithm which is most suitable for this data set is K-Means. The reason for choosing K-Means is:

1. The data visualization shows clear distinct clusters for data set.
2. The visualization also helps us know about the optimal number of clusters in for our data set.

These two key insights second the K-Mean Clustering Algorithm as choice of algorithm to be used for this data set.

References

Nathan Landman, C. W., Hannah Pang. (2017). K-means clustering. *Brilliant.org*.

Kilitcioglu, D. (2018). Hierarchical clustering and its applications. *TowardsDataScience.com*.

Forsyth, D. (2019). Clustering, In *Applied machine learning*. Springer.

Petrovic, S. (2006). A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters, In *Proceedings of the 11th nordic workshop of secure it systems*. Citeseer.

Pezzotti, N., Thijssen, J., Mordvintsev, A., Höllt, T., Van Lew, B., Lelieveldt, B. P., Eisemann, E., & Vilanova, A. (2019). Gpgpu linear complexity t-sne optimization. *IEEE transactions on visualization and computer graphics*, *26*(1), 1172–1181.

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Song, W., Wang, L., Liu, P., & Choo, K.-K. R. (2019). Improved t-sne based manifold dimensional reduction for remote sensing data processing. *Multimedia Tools and Applications*, *78*(4), 4311–4326.

Li, W., Cerise, J. E., Yang, Y., & Han, H. (2017). Application of t-sne to human genetic data. *Journal of bioinformatics and computational biology*, *15*(04), 1750017.

Violante, A. (2018). An introduction to t-sne with python example. *TowardsDataScience.com*.