# DMML6

February 23, 2025

[15]:

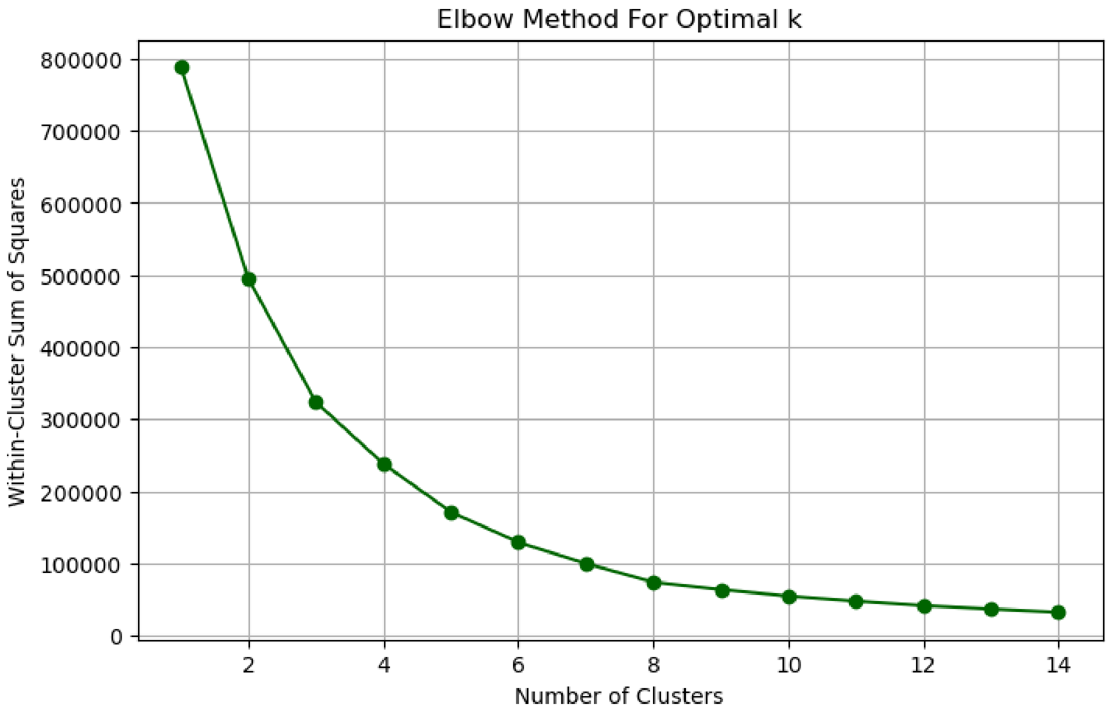
```python
# ==========================
# Step 1: Data Loading & Preprocessing for KMeans Clustering
# ==========================
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load dataset
dataset = pd.read_csv(r"C:
 ↪\Users\Karthikeyan\Documents\Data\Network\Midterm_53_group.csv")
print(dataset.head())

features = ["Time", "Length"]

# Encode categorical columns ("Source", "Destination", "Protocol")
for col in ["Source", "Destination", "Protocol"]:
    le = LabelEncoder()
    dataset[col] = le.fit_transform(dataset[col])

# Extract numerical features and scale them
x = dataset[features]
print(x)

scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
print(x_scaled)
```

```
       Time             Source  No.      Destination Protocol  Length  \
0  0.000000     192.167.8.166    1  192.167.255.255     NBNS      92
1  0.784682     192.167.8.166    2  192.167.255.255     NBNS      92
2  1.169060  VMware_8a:5c:e6    3        Broadcast      ARP      60
3  2.167949  VMware_8a:5c:e6    4        Broadcast      ARP      60
4  3.170095  VMware_8a:5c:e6    5        Broadcast      ARP      60

                                   Info
0                    Name query NB WPAD<00>
1                    Name query NB WPAD<00>
2  Who has 192.167.7.175? Tell 192.167.0.1
```

```
3  Who has 192.167.7.175? Tell 192.167.0.1
4  Who has 192.167.7.175? Tell 192.167.0.1
               Time  Length
0          0.000000      92
1          0.784682      92
2          1.169060      60
3          2.167949      60
4          3.170095      60
...             ...     ...
394131  1255.897236      98
394132  1255.897921      98
394133  1255.993209      74
394134  1256.921232      98
394135  1256.922008      98

[394136 rows x 2 columns]
[[-2.96506256 -1.06712294]
 [-2.9620858  -1.06712294]
 [-2.96062763 -1.10533781]
 ...
 [ 1.79965264 -1.0886188 ]
 [ 1.80317318 -1.05995765]
 [ 1.80317612 -1.05995765]]
```

[17]:

```python
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 15):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42, n_init=10)
    kmeans.fit(x_scaled)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(range(1, 15), wcss, marker='o', color='darkgreen')
plt.title("Elbow Method For Optimal k")
plt.xlabel("Number of Clusters")
plt.ylabel("Within-Cluster Sum of Squares")
plt.grid(True)
plt.show()
```



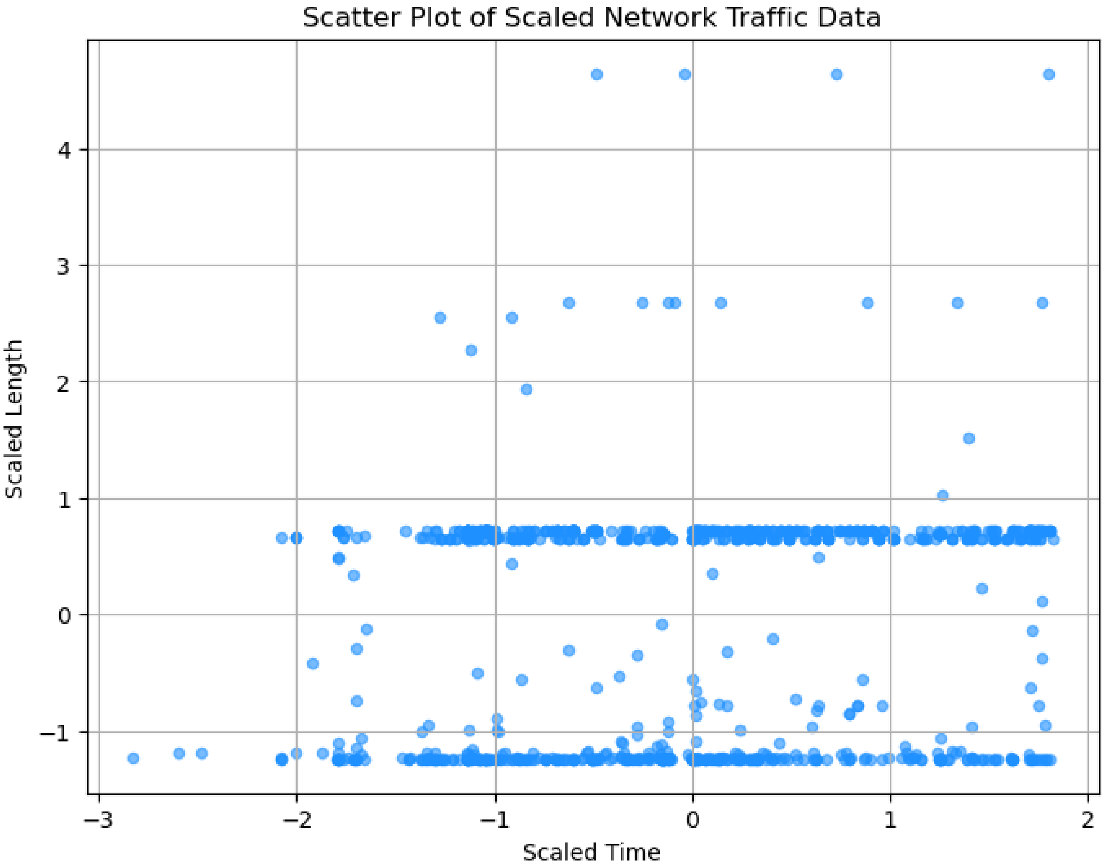
[23]:

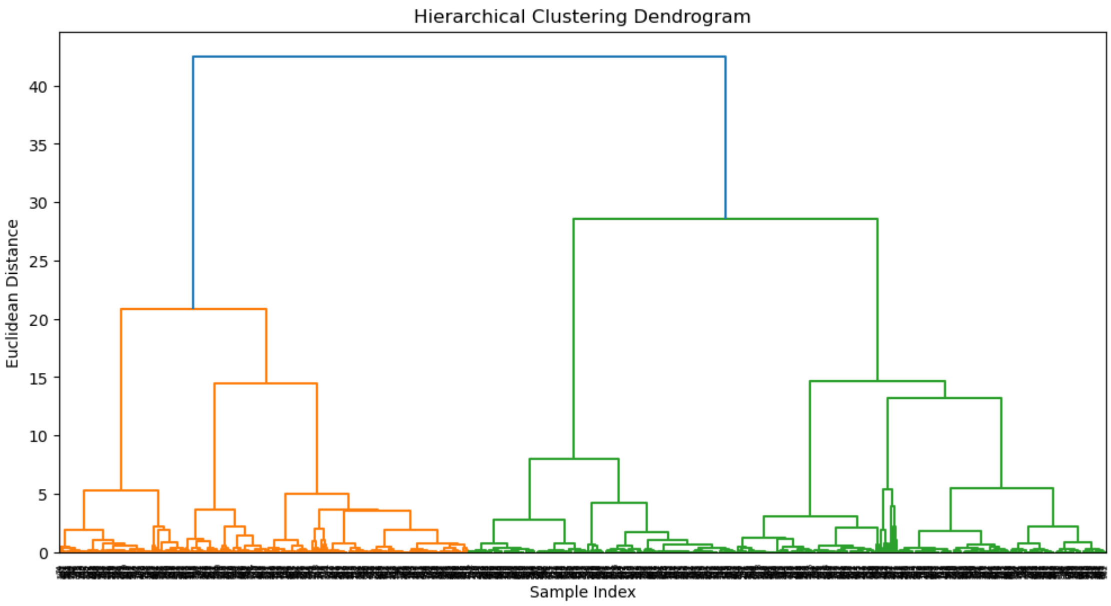
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
import scipy.cluster.hierarchy as sch

# Reload dataset
dataset = pd.read_csv(r"C:
 ↪\Users\Karthikeyan\Documents\Data\Network\Midterm_53_group.csv")
print(dataset.head())

features = ["Time", "Length"]

# Take a random sample of 1000 rows for computational ease
dataset_sample = dataset.sample(n=1000, random_state=42)

# Encode categorical columns ("Source", "Destination", "Protocol")
label_encoders = {}
for col in ["Source", "Destination", "Protocol"]:
    le = LabelEncoder()
    dataset[col] = le.fit_transform(dataset[col])
    label_encoders[col] = le

# Extract the sample features
```

```python
x_sample = dataset_sample[features]
print(x_sample)

# ==============================
# Graphical Representation
# ==============================

# 1. Scatter Plot of Scaled Features
scaler = StandardScaler()
x_sample_scaled = scaler.fit_transform(x_sample)

plt.figure(figsize=(8,6))
plt.scatter(x_sample_scaled[:, 0], x_sample_scaled[:, 1],
            color='dodgerblue', alpha=0.6, s=20)
plt.title("Scatter Plot of Scaled Network Traffic Data")
plt.xlabel("Scaled Time")
plt.ylabel("Scaled Length")
plt.grid(True)
plt.show()

# 2. Hierarchical Clustering Dendrogram
# Compute the linkage matrix using the Ward method
linkage_matrix = sch.linkage(x_sample_scaled, method='ward')

plt.figure(figsize=(12,6))
dendrogram = sch.dendrogram(linkage_matrix,
                            color_threshold=0.7 * max(linkage_matrix[:, 2]))
plt.title("Hierarchical Clustering Dendrogram")
plt.xlabel("Sample Index")
plt.ylabel("Euclidean Distance")
plt.show()
```

```
       Time           Source  No.      Destination Protocol  Length  \
0  0.000000    192.167.8.166    1  192.167.255.255     NBNS      92   
1  0.784682    192.167.8.166    2  192.167.255.255     NBNS      92   
2  1.169060  VMware_8a:5c:e6    3        Broadcast      ARP      60   
3  2.167949  VMware_8a:5c:e6    4        Broadcast      ARP      60   
4  3.170095  VMware_8a:5c:e6    5        Broadcast      ARP      60   

                                      Info  
0                   Name query NB WPAD<00>  
1                   Name query NB WPAD<00>  
2  Who has 192.167.7.175? Tell 192.167.0.1  
3  Who has 192.167.7.175? Tell 192.167.0.1  
4  Who has 192.167.7.175? Tell 192.167.0.1  
               Time  Length
351660  1159.739058      54
147074   684.013951     580
```

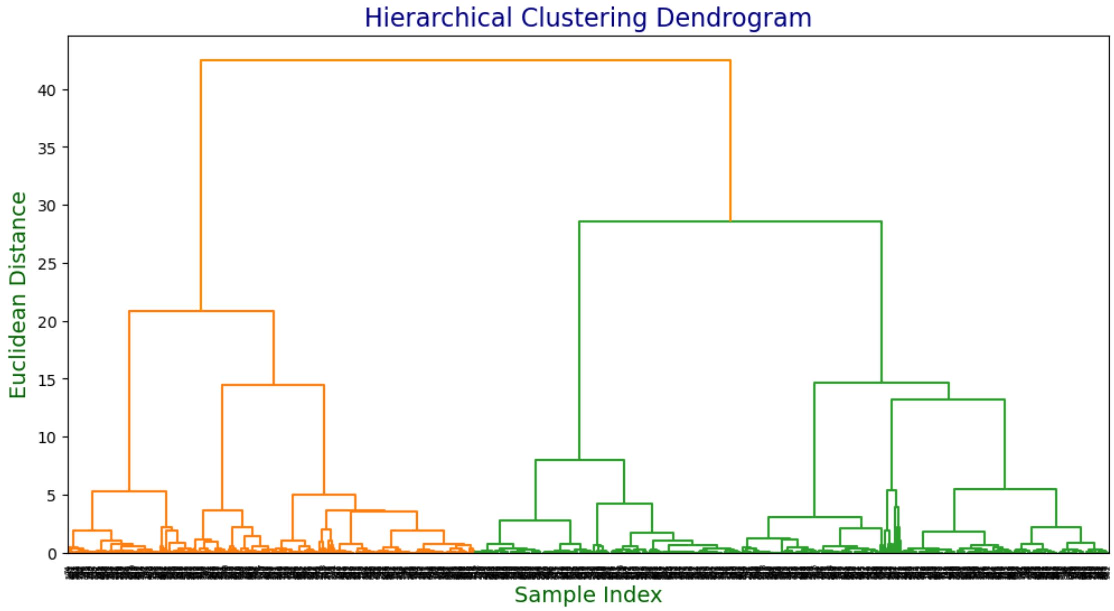
```
141496   653.704734    1514
224466   814.918402     405
381701  1233.302231    1514
...             ...     ...
372835  1223.674828    1514
297389   988.107413      60
279611   941.512538    1462
104787   571.123548      54
508      103.362516      98

[1000 rows x 2 columns]
```





[25]:
```
import scipy.cluster.hierarchy as sch
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Scale the sample data
scaler = StandardScaler()
x_sample_scaled = scaler.fit_transform(x_sample)
print(x_sample_scaled)

plt.figure(figsize=(12, 6))
# Compute the linkage matrix using the Ward method
linkage_matrix = sch.linkage(x_sample_scaled, method='ward')

# Plot dendrogram with updated color settings:
# - 'above_threshold_color' is set to 'darkorange' for branches above the␣
↪threshold.
dendrogram = sch.dendrogram(linkage_matrix,
                            color_threshold=0.7 * max(linkage_matrix[:, 2]),
                            above_threshold_color='darkorange')

plt.title("Hierarchical Clustering Dendrogram", fontsize=16, color='navy')
plt.xlabel("Sample Index", fontsize=14, color='darkgreen')
plt.ylabel("Euclidean Distance", fontsize=14, color='darkgreen')
plt.show()
```

```
[[ 1.46000345 -1.24068847]
 [-0.36628354 -0.53400222]
[-0.48263925  0.7208361 ]
...
[ 0.62224179  0.65097358]
[-0.79966465 -1.24068847]
[-2.59537792 -1.18157403]]
```



```
[ ]:
```