Network Traffic Classification

1. using K-Means Clustering :

#Import all required libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

 # Load the dataset

```
dataset = pd.read_csv('/content/drive/MyDrive/DMML/nw2.csv')
dataset.head()
features = ["Time","Length"]
```

#Encode the data

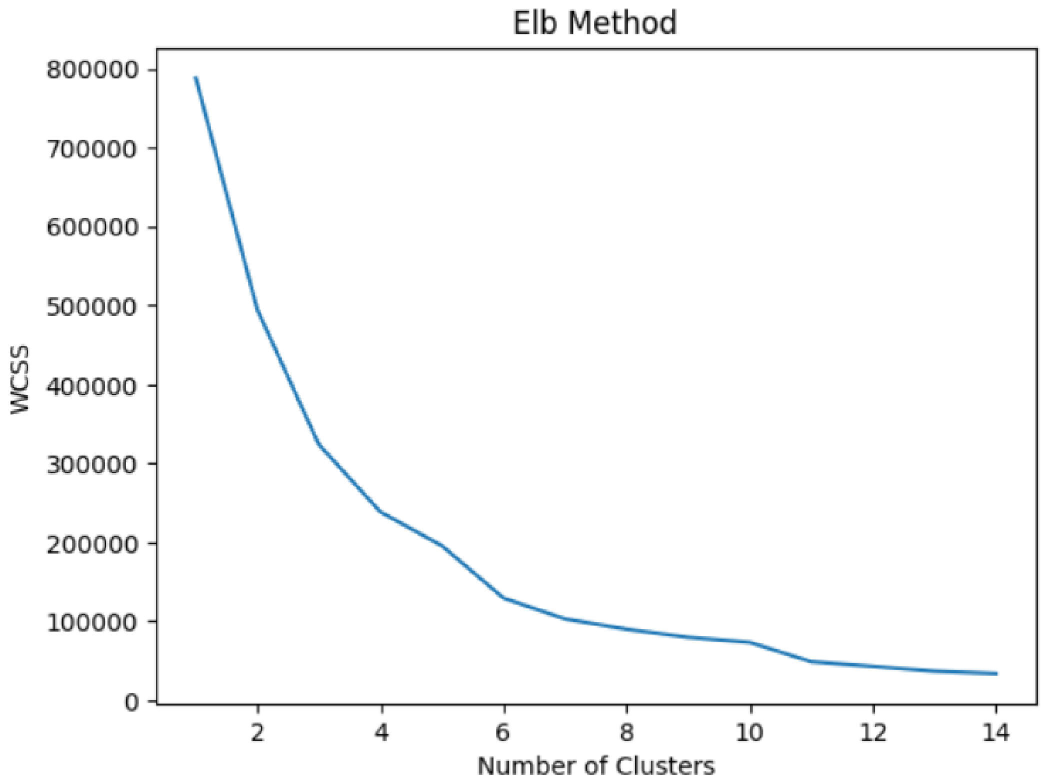
```
label_encoders = {}
for col in ["Source", "Destination", "Protocol"]:
    le = LabelEncoder()
    dataset[col] = le.fit_transform(dataset[col])
    label_encoders[col] = le
X = dataset[features]
print(X)
```

```
               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]
```

```
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
print(X_scaled)
```

```
[[-2.96506256 -1.06712294]
 [-2.9620858  -1.06712294]
 [-2.96062763 -1.10533781]
 ...
 [ 1.79965264 -1.0886188 ]
 [ 1.80317318 -1.05995765]
 [ 1.80317612 -1.05995765]]
```

```
#Compute within-cluster sum of squares (WCSS) for different cluster sizes
from sklearn.cluster import KMeans
wcss= []
for i in range(1,15):
  kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
  kmeans.fit(X_scaled)
  wcss.append(kmeans.inertia_)
#Plot the elbow method
plt.plot(range(1,15),wcss)
plt.title("Elb Method")
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```

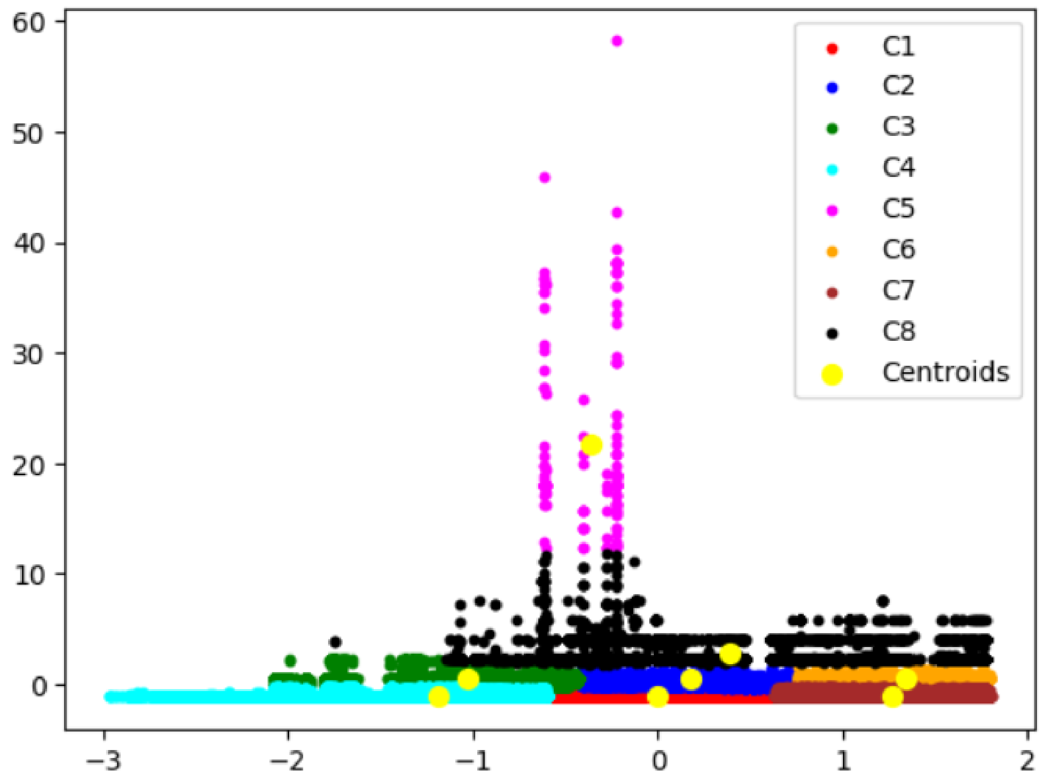


#K-Means clustering and visualize the model.

```
kmeans = KMeans(n_clusters=8, random_state=42, n_init=10)

y_kmeans = kmeans.fit_predict(X_scaled)

print(np.unique(y_kmeans))

plt.scatter(X_scaled[y_kmeans==0,0],X_scaled[y_kmeans==0,1],s=10,c='red',label='C1')

plt.scatter(X_scaled[y_kmeans==1,0],X_scaled[y_kmeans==1,1],s=10,c='blue',label='C2')

plt.scatter(X_scaled[y_kmeans==2,0],X_scaled[y_kmeans==2,1],s=10,c='green',label='C3')

plt.scatter(X_scaled[y_kmeans==3,0],X_scaled[y_kmeans==3,1],s=10,c='cyan',label='C4')

plt.scatter(X_scaled[y_kmeans==4,0],X_scaled[y_kmeans==4,1],s=10,c='magenta',label='C5')

plt.scatter(X_scaled[y_kmeans==5,0],X_scaled[y_kmeans==5,1],s=10,c='orange',label='C6')
```

```
plt.scatter(X_scaled[y_kmeans==6,0],X_scaled[y_kmeans==6,1],s=10,c='brown',label='C7')

plt.scatter(X_scaled[y_kmeans==7,0],X_scaled[y_kmeans==7,1],s=10,c='black',label='C8')

# plt.scatter(X[y_kmeans==3]['Protocol'],X[y_kmeans==3]['Length'],s=100,c='cyan',label='C4')

# plt.scatter(X[y_kmeans==4]['Protocol'],X[y_kmeans==4]['Length'],s=100,c='magenta',label='C5')

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1],s=50,c='yellow',label='Centroids')

plt.legend()
```

```
[0 1 2 3 4 5 6 7]
<matplotlib.legend.Legend at 0x7afdbae27dd0>
```



2. using Hierarchical Clustering:

#Import all required libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
import scipy.cluster.hierarchy as sch
#Loading and sampling of the dataset
dataset = pd.read_csv('/content/drive/MyDrive/DMML/nw2.csv')
dataset_sample = dataset.sample(n=1000, random_state=42)
dataset_sample.head()
features = ["Time","Length"]
#Perform label encoding
label_encoders = {}
for col in ["Source", "Destination", "Protocol"]:
    le = LabelEncoder()
    dataset[col] = le.fit_transform(dataset[col])
    label_encoders[col] = le
X = dataset_sample[features]
print(X)
```

```
                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]
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
print(X_scaled)
plt.figure(figsize=(12, 6))
dendrogram = sch.dendrogram(sch.linkage(X_scaled, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Time')
plt.ylabel('Length')
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]]
```