Etape 1 : Importer les packages nécessaires et de la dataset

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


traffic_data = pd.read_csv('UPL_traffic_reseau.csv', encoding =
'unicode_escape')


traffic_data.head()


traffic_data.shape


traffic_data.columns
```

Étape 2 : Étape de prétraitement des données

```
traffic_data.info()


traffic_data['Source'].unique()


traffic_data['Source'].value_counts()


traffic_data['Destination'].value_counts()


traffic_data['Protocol'].unique()


traffic_data.head()


traffic_data.drop(['No.','Time', 'Source','Destination','Info'],
axis='columns', inplace=True)
```

```
traffic_data


traffic_data.drop('Protocol', axis=1, inplace=True)


traffic_data.head()


traffic_data.isnull().sum()


traffic_data['TCP Segment Len'].fillna(traffic_data['TCP Segment
Len'].mean(), inplace=True)
traffic_data['Stream index'].fillna(traffic_data['Stream
index'].mean(), inplace=True)


traffic_data.isnull().sum()


traffic_data.head()


traffic_data.head()


traffic_data['Stream index'].max()


traffic_data['Frame length on the wire'].max()


traffic_data.shape


x = traffic_data.iloc[0:1500,2:].values
x
```

Étape 3 : Trouver le nombre optimal de clusters à l'aide de la méthode du coude

Method Elbow

```
# Trouver le nombre optimal de clusters en utilisant la methode du coude
from sklearn.cluster import KMeans


wcss_list = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss_list)
plt.title('Le graphique de la methode Elbow')
plt.xlabel('Nombre de clusters (k)')
plt.ylabel('wcss_list')
plt.show()
```

Étape 4 : Entraînement de l'algorithme K-means sur les données d'entraînement

```
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)


kmeans.fit(x)


y_predict = kmeans.predict(x)


y_predict


#Visualisation des clusters
plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Groupe 1') # Pour le premier groupe.
plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Groupe 2') # Pour le deuxième groupe.
plt.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Groupe 3') # Pour le troisième groupe.
```

```
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s = 300, c = 'yellow', label = 'Centroïde')
plt.title('Groupes de Packet.')
plt.xlabel('Stream index')
plt.ylabel('Frame length on the wire')
plt.legend()
plt.show()
```