A

1)

Can our customers be grouped into separate categories using k-means clustering so we may target them with different advertising and promotional campaigns?

2

One goal would be to define at least two groups of customers from the churn data set using k-means clustering.

B

1)

K-means clustering analyzes the data set by partitioning a data set into a specified number of clusters. Cluster centers are randomly chosen from points in the data set. All other data points are then assigned to the cluster that is closest to it. Cluster centers are recalculated using the means of the data points assigned to it. Data points are assigned to the clusters again. This process can repeat many times. An expected outcome is that all data points will be assigned to a cluster and the clusters will have minimum inertia when the algorithm has finished.

2)

One assumption of k-means clustering is that all data points should be numerical and continuous.

3) numpy

This is used for working with numpy arrays that are returned from the scaling function.

matplotlib.pyplot

This is used for visualizing inertia using the elbow method to find the optimal number of clusters.

sklearn.cluster import KMeans

This is the actual clustering algorithm that creates the model to cluster our data.

from sklearn.preprocessing import StandardScaler

   This will be used to normalize the continuous variables.

C

1)

    One data preprocessing goal is to scale the variables before clustering them.

2) The inital data set variables will be:

    City categorical,
    County categorical,
    Area categorical,
    Marital categorical,
    Gender categorical,
    Churn categorical,
    Techie categorical,
    StreamingTV categorical,
    Multiple categorical

   Children continuous,
   Age continuous,
   Income continuous,
   Outage_sec_perweek continuous,
   MonthlyCharge continuous,
   Bandwidth_GB_year continuous

In [ ]:

3)

prepare data

read in data and drop index column.

In [1]:
```
import pandas as pd
# Assuming your CSV file is named 'data.csv', adjust the file path as needed
file_path = '/home/dj/skewl/D212/1/churn_clean.csv'
```

```
pd.set_option('display.max_columns', None)
# Read the data from the CSV file into a DataFrame
df = pd.read_csv(file_path)
#drop index column
df = df.loc[:, ~df.columns.str.contains('Unnamed')]
```

check for missing values.

In [2]:
```
# Identify missing values using isna() method
missing_values = df.isna().sum()
# Print DataFrame with True for missing values and False for non-missing values
print(missing_values)
# no missing values.
```

```
CaseOrder              0
Customer_id            0
Interaction            0
UID                    0
City                   0
State                  0
County                 0
Zip                    0
Lat                    0
Lng                    0
Population             0
Area                   0
TimeZone               0
Job                    0
Children               0
Age                    0
Income                 0
Marital                0
Gender                 0
Churn                  0
Outage_sec_perweek     0
Email                  0
Contacts               0
Yearly_equip_failure   0
Techie                 0
Contract               0
Port_modem             0
Tablet                 0
InternetService        0
Phone                  0
Multiple               0
OnlineSecurity         0
OnlineBackup           0
DeviceProtection       0
TechSupport            0
StreamingTV            0
StreamingMovies        0
PaperlessBilling       0
PaymentMethod          0
Tenure                 0
MonthlyCharge          0
Bandwidth_GB_Year      0
Item1                  0
Item2                  0
Item3                  0
Item4                  0
Item5                  0
Item6                  0
```

```
Item7                  0
Item8                  0
dtype: int64
```

separate continuous and categorical variables. Then one hot encode the categorical variables

In [3]:
```
#split continuous and categorical variables into separate dataframes
dfcon = df[['Age','Income','Bandwidth_GB_Year','Age','MonthlyCharge']]
dfcat = df[['Gender','Area','City','County','Marital','Churn','Techie','StreamingTV','Multiple']]
#one-hot encode categorical data and drop first level of each
dfcat_encoded = pd.get_dummies(dfcat,drop_first=True)
```

Concatenate encoded categorical variables with continuous variables. Write prepared data to file. Normalize all variables and put them in a numpy array.

In [4]:
```
#normalize data after encoding
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
#concatenate the columns
df = pd.concat([dfcon, dfcat_encoded], axis=1)
#write the prepared data to .csv file
df.to_csv('prepared-data.csv', index=False)
# scale the data frame
df = scaler.fit_transform(df)
```

D

1)

I determined that 5 is the optimal number of clusters using the elbow method. This means that adding more than 5 clusters does not significantly decrease the inertia or within cluster sum of squares variance within the clusters.

2)

Code to plot the inertia of the clusters using the elbow method.

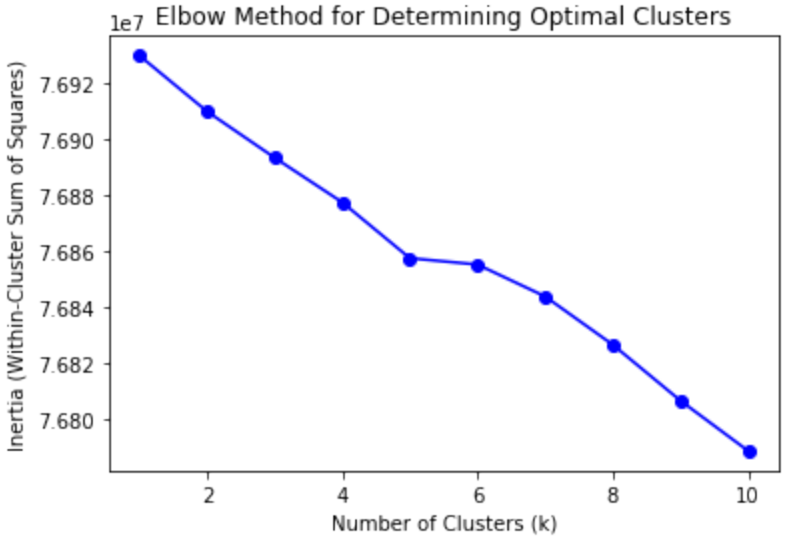
In [5]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Define a range of cluster numbers to test
k_values = range(1, 11)  # Test cluster numbers from 1 to 10
```

```
inertiaArray = []

# Calculate inertia for each `k` value
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df)
    inertiaArray.append(kmeans.inertia_)

# Plot inertia to find the "elbow"
plt.plot(k_values, inertiaArray, 'bo-')  # 'bo-' indicates blue circles with lines
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia (Within-Cluster Sum of Squares)")
plt.title("Elbow Method for Determining Optimal Clusters")
plt.show()
```



E

1)

The quality of the clusters that were created are being evaluated by the metric called inertia. Inertia is also called the within cluster sum of squares. This is calculated by summing the squared distance from each data point and it's cluster center or centroid. Lower inertia indicates more compact clusters. K-means algorithm tries to produce clusters with a low inertia. With a cluster number set to k=5 we get an inertia of approximately 7.686.

2)

The results of the cluster analysis show that the customers in the churn data set can be grouped into 5 groups. The analysis shows that grouping the customers into more than 5 groups does not significantly reduce the inertia of each cluster. This analysis shows us that our customers can be categorized into 5 different groups. This can be useful for targeting these different groups for promotional or advertising campaigns.

3)

One limitation of the data analysis is that it may be hard to discern what is different about the customers in each of the 5 clusters. While we know that there are 5 groups of customers that are statistically different from each other, it is unknown if the differences are meaningful, or the groups can be targeted separately because of their cluster membership. From this analysis we don't know what exactly defines each cluster and makes it different from the others. Further analysis of each cluster is needed before we can make actionable changes based on k means clustering.

4)

One course of action based on the results of this analysis would be that we should spend more time and resources analyzing the customers in each of these 5 clusters. Now that we know our customers fall into 5 different groups we can tailor our marketing, promotional, and advertising campaigns to suit customers in these categories. We should analyze each of the 5 clusters and find out what makes that cluster different. When that is known the marketing team can develop a plan to appeal to each of the 5 clusters.

In [ ]: