D212 – Assessment OFM3 TASK 1: CLUSTERING TECHNIQUES

Part I: Research Question

A. Describe the purpose of this data mining report by doing the following:

1. Propose **one** question relevant to a real-world organizational situation that you will answer using **one** of the following clustering techniques:
   - *k*-means
   - hierarchical

Which customer features provided in the churn dataset are more relevant to understand customer behavior and might indicate that customers will churn? The clustering technique can help us better understand customer behavior and identify patterns in churn. I will be using the k-means clustering technique.

2. Define **one** goal of the data analysis. Ensure that your goal is reasonable within the scope of the scenario and is represented in the available data.

The company's stakeholders will be able to determine which features are more determinant in customer churn. With that information, stakeholders will be able to create/modify marketing campaigns in order to retain more customers and successfully acquire new ones with the right service package.

Part II: Technique Justification

B. Explain the reasons for your chosen clustering technique from part A1 by doing the following:

1. Explain how the clustering technique you chose analyzes the selected dataset. Include expected outcomes.

With the K-means clustering technique, we are trying to identify patterns in our dataset. The algorithm identifies k number of centroids (the centroid represents the center of the cluster and it changes in each step of the iteration), and allocates datapoints to the nearest centroids (minimum Euclidean distance), always trying to keep the centroids as small as possible[1]. In the end, we will be able to identify similar groups of customers.

Centroid: $C_i = \frac{1}{||Si||} \sum_{xj \in Si} \quad xj$

C_i represents the i-th centroid

S_i all points in set_i with centroid C_i

x_j is the j-th point from the set_i

$||S_i||$ number of points in set_i

The idea of the K-Means algorithm is to find k centroid points (C_1, C_2, . . . C_k) by minimizing the sum over each cluster of the sum of the square of the distance between the point and its centroid[4].

In our case, we are going to use K-means++, which is a smart centroid initialization technique[4].

This algorithm works as follows:
1-) A random centroid is chosen (C_1);
2-) A distance of all points in the dataset is calculated from that chosen centroid (calculate the distance, d_i, of x_i from the farthest centroid)
3-) Choose a new point as the new centroid, x_i, that has maximum probability proportional to d_i
4-) Repeat steps 2 and 3 until k-centroids are found.

2. Summarize **one** assumption of the clustering technique.

One important assumption of this technique is that each cluster has a roughly equal number of observations[2].

3. List the packages or libraries you have chosen for Python or R, and justify how *each* item on the list supports the analysis.

I have been working with Python since D206 and will continue learning this language. I have been working with the free version of Pycharm (Community). The packages and libraries are:

-) Pandas: package to read our dataset, present some statistics of the features, clean and modify data
-) Numpy: performs a wide variety of mathematical operations on arrays
-) Matplotlib: important package in any data science project. It helps with data visualization.
-) Scikit-learn: packages that will split, test, train and make predictions on our dataset
-) Seaborn: this package will be used for more detailed graphs and matrices
-) Scipy: library for the k-means algorithm

Part III: Data Preparation

C. Perform data preparation for the chosen dataset by doing the following

1. Describe **one** data preprocessing goal relevant to the clustering technique from part A1.

K-means requires numerical data only, with no noise and outliers. So for this case, we are leaving out all categorical data and making boxplot visualizations of our numerical data to identify possible anomalies.

2. Identify the initial dataset variables that you will use to perform the analysis for the clustering question from part A1, and label *each* as continuous or categorical.

All features will be included in the analysis except for: “CaseOrder”, “CustomerID”, “Interaction” and “UID.”

All customer demographics will be included: **Categorical**: City, State, County, TimeZone, Job, Area, Marital, Gender;

Continuous: Zip, Lat, Ln, Population, Children, Age, Income.

Churn: categorical target variable, **y**

Continuous: Outage_sec_perweek, Email, Contacts, Yearly_equip_failure, Tenure, MonthlyCharge, Bandwidth_GB_Year.

Categorical: Techie, Contract, Port_modem, Tablet, InternetService, Phone, Multiple, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport**,** StreamingTV, StreamingMovies, PaperlessBilling, PaymentMethod

Discrete Ordinal Predictor Variables: Item1 to Item8. The last few columns contain customer survey data.

3. Explain *each* of the steps used to prepare the data for the analysis. Identify the code segment for *each* step.

1-) Read the provided dataset by WGU with pandas
2-) Evaluation of the dataset using info, shape, describe(), types
3-) Check for misspellings and missing data (if there is any missing data replace the missing data with meaningful measures of central tendency – median or mean for numerical and mode for categorical)
4-) Exclude variables that are not important for our analysis
5-) Extract final dataset

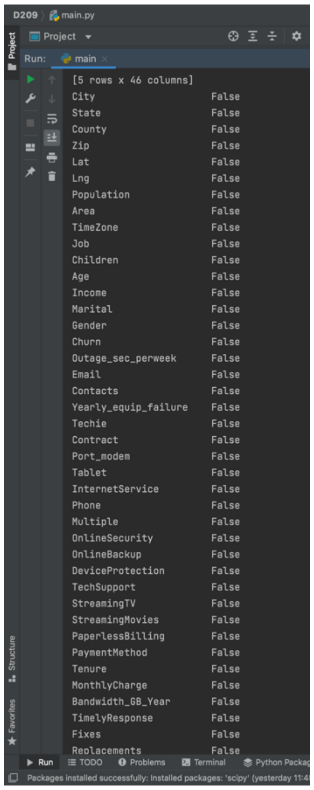
Code:

```
# Importing Libraries
import pandas as pd

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

# Libraries for Visualization Purposes
import matplotlib.pyplot as plt
# from scipy.spatial.distance import cdist

#Scipy
from scipy.cluster.vq import kmeans, vq
```

```
# Loading the Churn Dataset
churn_df = pd.read_csv('/Users/bia/Desktop/Datasets for
D212/churn_clean.csv')

#Dataset Size with SHAPE
print(churn_df.shape)

#Dataset columns
print(churn_df.columns)

#Dataset Info
print(churn_df.info)

#Dataset Columns Types
print(churn_df.dtypes)

#Basic Stats of the data
print(churn_df.describe())

#Removing some unecessary columns
churn_df = churn_df.drop(columns=['CaseOrder', 'Customer_id', 'Interaction',
'UID'])

#Checking if my changes were in place
print(churn_df.head())
```

Checking for missing data:

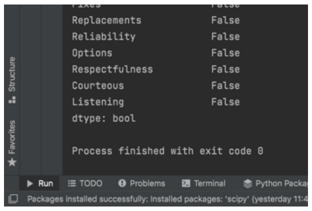
```
#Finding missing values in my dataset
churn_df.isnull().any(axis=1)
null_values = churn_df.isna().any()
print(null_values)
```



```
[5 rows x 46 columns]
City                    False
State                   False
County                  False
Zip                     False
Lat                     False
Lng                     False
Population              False
Area                    False
TimeZone                False
Job                     False
Children                False
Age                     False
Income                  False
Marital                 False
Gender                  False
Churn                   False
Outage_sec_perweek      False
Email                   False
Contacts                False
Yearly_equip_failure    False
Techie                  False
Contract                False
Port_modem              False
Tablet                  False
InternetService         False
Phone                   False
Multiple                False
OnlineSecurity          False
OnlineBackup            False
DeviceProtection        False
TechSupport             False
StreamingTV             False
StreamingMovies         False
PaperlessBilling        False
PaymentMethod           False
Tenure                  False
MonthlyCharge           False
Bandwidth_GB_Year       False
TimelyResponse          False
Fixes                   False
Replacements            False
```

```
Packages installed successfully: Installed packages: 'scipy' (yesterday 11:4
```

Figure 1: Checking for Missing Data - I



```
Replacements            False
Reliability             False
Options                 False
Respectfulness          False
Courteous               False
Listening               False
dtype: bool

Process finished with exit code 0
```

```
Packages installed successfully: Installed packages: 'scipy' (yesterday 11:4
```

Figure 2: Checking for Missing Data – II

As we can see in Figure 1 and Figure 2, all features return "False" for missing data.

4. Provide a copy of the cleaned dataset: prepared_churn_data_kmeans.xls will be uploaded with the video recording and document.

```
#Extract the "Prepared"" dataset
churn_num.to_csv('prepared_churn_data_kmeans.csv')
churn_num = pd.read_csv('prepared_churn_data_kmeans.csv')
df = churn_num.columns
print('The dataset columns are ', df)
```

Part IV: Analysis

D. Perform the data analysis and report on the results by doing the following:
   1. Describe the analysis technique you used to appropriately analyze the data. Include screenshots of the intermediate calculations you performed.

I start my analysis by creating histograms for the numeric data:

```
#Histogram for Numerical Data
dataset = churn_df[['MonthlyCharge', 'Bandwidth_GB_Year',
'Yearly_equip_failure', 'Children', 'Age', 'Income',
                    'Outage_sec_perweek','Email', 'Contacts', 'Tenure']]
fig = plt.figure(figsize=(15, 12))
plt.suptitle('Histograms - Numerical Data \n', horizontalalignment="center",
fontstyle="normal", fontsize=24,
             fontfamily="sans-serif")
for i in range(dataset.shape[1]):
    plt.subplot(6, 3, i + 1)
    f = plt.gca()
    f.set_title(dataset.columns.values[i])
    vals = np.size(dataset.iloc[:, i].unique())
    if vals >= 100:
        vals = 100
    plt.hist(dataset.iloc[:, i], bins=vals, color='#ec838a')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

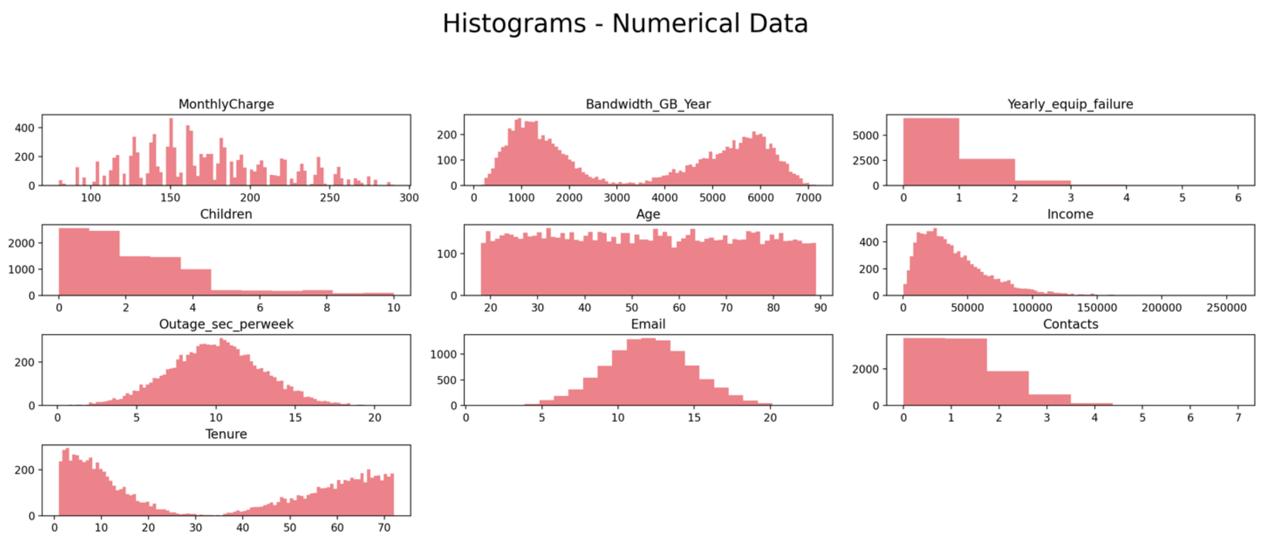


Figure 3: Histograms - Numerical Data

The second step of the data analysis is to create a scatter matrix using the features above to understand better the relationships in between them.

```
#Scatter Matrix
churn_num = churn_df[['MonthlyCharge', 'Bandwidth_GB_Year',
'Yearly_equip_failure', 'Children', 'Age', 'Income',
                          'Outage_sec_perweek','Email', 'Contacts', 'Tenure']]
scatter_matrix = pd.plotting.scatter_matrix(churn_num, figsize = [15,15],
diagonal = "kde", color = "magenta")

for ax in scatter_matrix.ravel():
    ax.set_xlabel(ax.get_xlabel(), fontsize = 10, rotation = 90)
    ax.set_ylabel(ax.get_ylabel(), fontsize = 10, rotation = 0)
plt.title('Scatter Matrix')
plt.show()
```

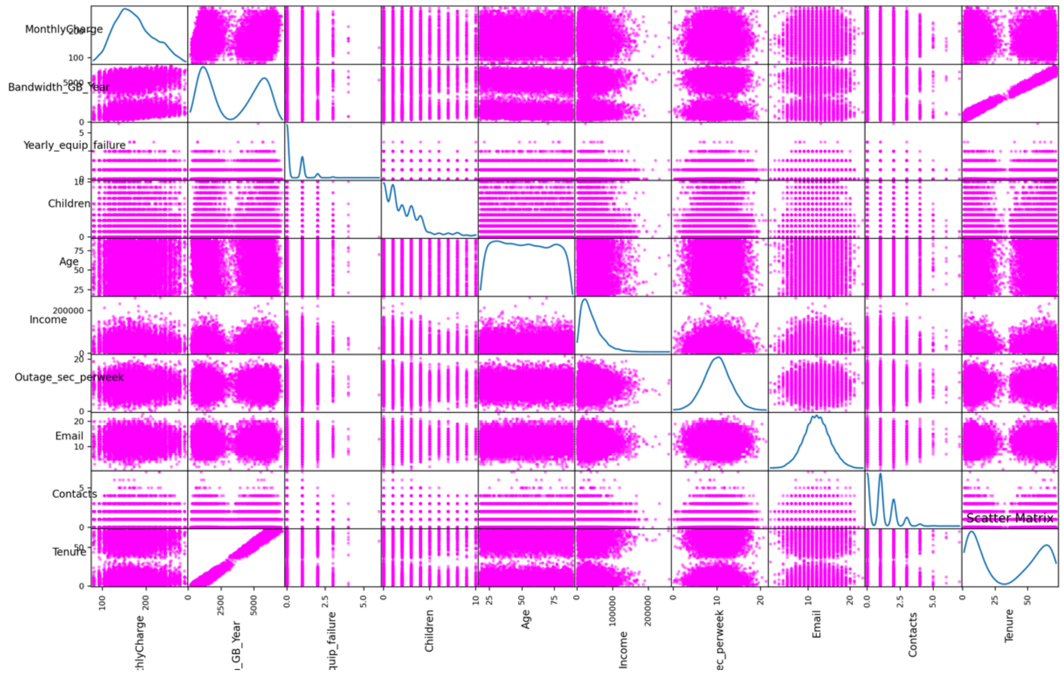


Figure 4: Scatter Matrix

The second row on the right side of the picture shows a very clear relationship between the "Tenure" and "Bandwidth_GB_Year" features.

Creating a heatmap will help to analyze the relationships:

```
#Create dataframe for heatmap bivariate analysis of correlation
churn_bivariate = churn_df[['MonthlyCharge', 'Bandwidth_GB_Year', 'Tenure',
'Outage_sec_perweek', 'Income']]
sns.heatmap(churn_bivariate.corr(), annot=True)
plt.show()
```

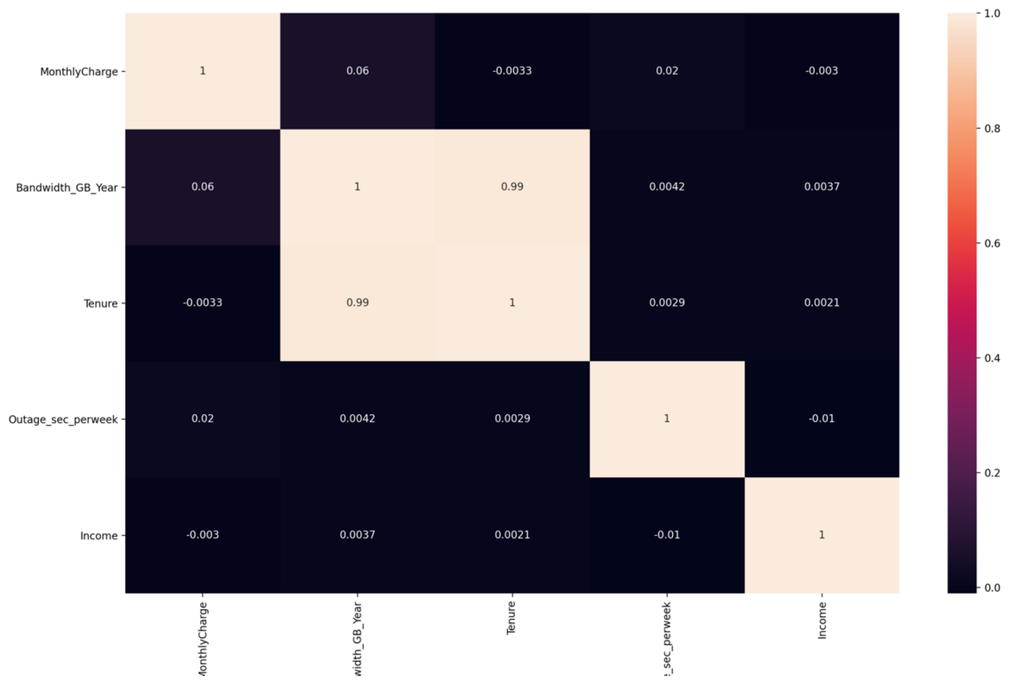


Figure 5: Heatmap

Checking for outliers:

```
# Create Seaborn boxplots for continuous variables
sns.boxplot('MonthlyCharge', data = churn_df)
plt.show()
print("MonthlyCharge Stats are: ", churn_df.MonthlyCharge.describe())

# Create Seaborn boxplots for continuous variables
sns.boxplot('Bandwidth_GB_Year', data = churn_df)
plt.show()
print("Bandwidth_GB_Year Stats are: ", churn_df.Bandwidth_GB_Year.describe())

# Create Seaborn boxplots for continuous variables
sns.boxplot('Age', data = churn_df)
plt.show()
print("Age Stats are: ", churn_df.Age.describe())

# Create Seaborn boxplots for continuous variables
sns.boxplot('Tenure', data = churn_df)
plt.show()
print("Tenure Stats are: ", churn_df.Tenure.describe())
```

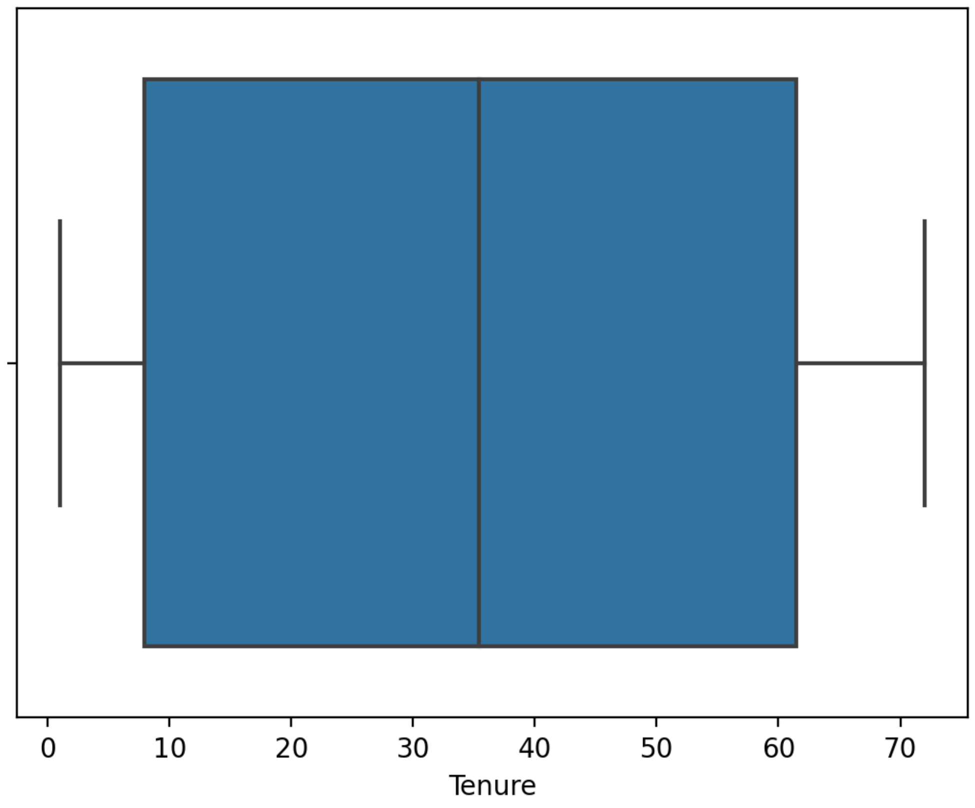


Figure 6: Tenure Boxplot

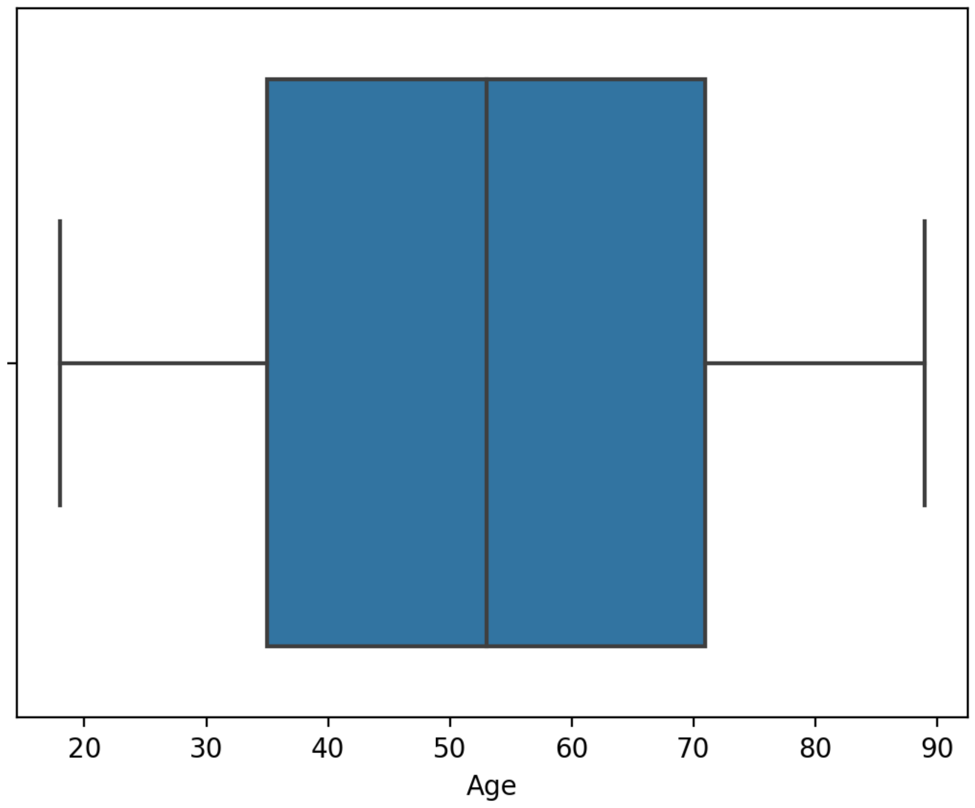


Figure 7: Age Boxplot

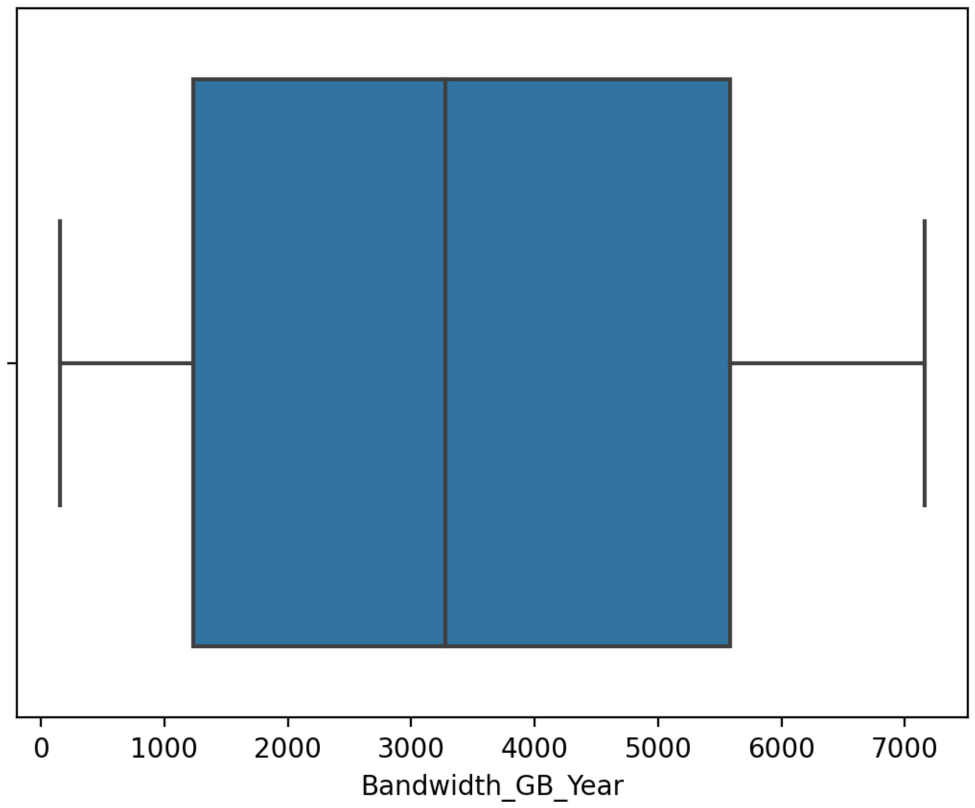


Figure 8: Bandwidth_GB_Year Boxplot

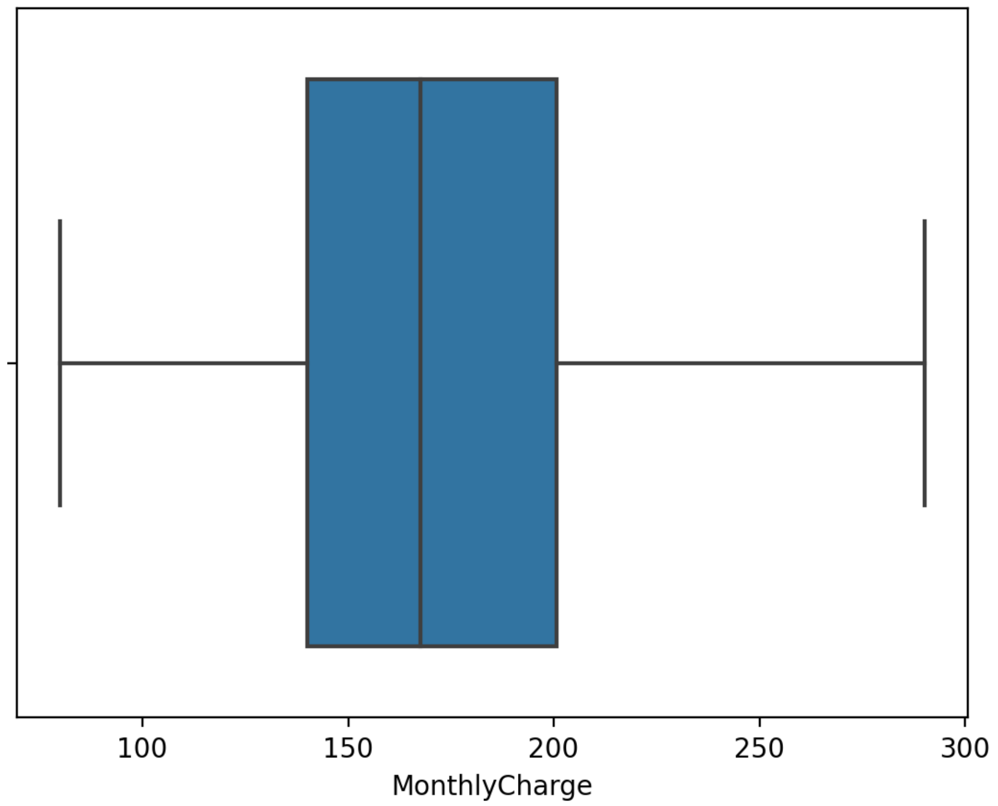


Figure 9: MonthlyCharge Boxplot

None of the continuous variables present outliers.

After this initial analysis has been done, we will start clustering analysis using the K-Means Method.

2. Provide the code used to perform the clustering analysis technique from part 2.

```
#Starting the KMeans Analysis
df = pd.read_csv('prepared_churn_data_kmeans.csv', index_col= 0)

#Importing KMeans
from sklearn.cluster import KMeans

#Selecting ggplot to make it look pretty
plt.style.use('ggplot')

#Selecting the features Tenure and MonthlyCharge (They are respectively
columns number 30 and 31)
X = churn_num.iloc[:,[2,1]].values
#Selecting the features Income and MonthlyCharge
# X = df.iloc[:,[4,1]].values
```

```
#Selecting the features Tenure and Bandwidth
# X = df.iloc[:,[2,3]].values

#Create a WCSS list
wcss = []

#Finding the optimal number of clusters
for i in range (1,10):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

#Plotting the optimal number
plt.plot(range(1,10), wcss)
plt.title('The Elbow Method Using Inertia')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()

#Tenure and Monthly Charge
# #Training the Method
# kmeans = KMeans(n_clusters = 6, init = 'k-means++', random_state = 42)
#
# y_kmeans = kmeans.fit_predict(X)
# print('Y_Kmeans are: ',y_kmeans)
#
# #Cluster Visualization
# plt.scatter(X[y_kmeans == 0,0], X[y_kmeans == 0,1], s = 10, c = 'red',
label = 'Cluster #1')
# plt.scatter(X[y_kmeans == 1,0], X[y_kmeans == 1,1], s = 10, c = 'blue',
label = 'Cluster #2')
# plt.scatter(X[y_kmeans == 2,0], X[y_kmeans == 2,1], s = 10, c = 'orange',
label = 'Cluster #3')
# plt.scatter(X[y_kmeans == 3,0], X[y_kmeans == 3,1], s = 10, c = 'magenta',
label = 'Cluster #4')
# plt.scatter(X[y_kmeans == 4,0], X[y_kmeans == 4,1], s = 10, c = 'green',
label = 'Cluster #5')
# plt.scatter(X[y_kmeans == 5,0], X[y_kmeans == 5,1], s = 10, c = 'cyan',
label = 'Cluster #6')
#
# #Plot the Centroids
# plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s =
100, c = 'yellow', label = 'Centroids')
# plt.xlabel('Tenure - months')
# plt.ylabel('MonthlyCharge - $')
# plt.title('6 Customer Clusters')
# plt.show()

# #Income and Monthly Charge
# #Training the Method
# kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
#
# y_kmeans = kmeans.fit_predict(X)
# print('Y_Kmeans are: ',y_kmeans)
#
```

```
# #Cluster Visualization
# plt.scatter(X[y_kmeans == 0,0], X[y_kmeans == 0,1], s = 10, c = 'red',
label = 'Cluster #1')
# plt.scatter(X[y_kmeans == 1,0], X[y_kmeans == 1,1], s = 10, c = 'blue',
label = 'Cluster #2')
# plt.scatter(X[y_kmeans == 2,0], X[y_kmeans == 2,1], s = 10, c = 'orange',
label = 'Cluster #3')
# plt.scatter(X[y_kmeans == 3,0], X[y_kmeans == 3,1], s = 10, c = 'magenta',
label = 'Cluster #4')
#
# #Plot the Centroids
# plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s =
100, c = 'yellow', label = 'Centroids')
# plt.xlabel('Income - $')
# plt.ylabel('MonthlyCharge - $')
# plt.title('4 Customer Clusters')
# plt.show()

#Tenure and Monthly Charge
# #Training the Method
# kmeans = KMeans(n_clusters = 6, init = 'k-means++', random_state = 42)
#
# y_kmeans = kmeans.fit_predict(X)
# print('Y_Kmeans are: ',y_kmeans)
#
# #Cluster Visualization
# plt.scatter(X[y_kmeans == 0,0], X[y_kmeans == 0,1], s = 10, c = 'red',
label = 'Cluster #1')
# plt.scatter(X[y_kmeans == 1,0], X[y_kmeans == 1,1], s = 10, c = 'blue',
label = 'Cluster #2')
# plt.scatter(X[y_kmeans == 2,0], X[y_kmeans == 2,1], s = 10, c = 'orange',
label = 'Cluster #3')
# plt.scatter(X[y_kmeans == 3,0], X[y_kmeans == 3,1], s = 10, c = 'magenta',
label = 'Cluster #4')
# plt.scatter(X[y_kmeans == 4,0], X[y_kmeans == 4,1], s = 10, c = 'green',
label = 'Cluster #5')
# plt.scatter(X[y_kmeans == 5,0], X[y_kmeans == 5,1], s = 10, c = 'cyan',
label = 'Cluster #6')
#
# #Plot the Centroids
# plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s =
100, c = 'yellow', label = 'Centroids')
# plt.xlabel('Tenure - months')
# plt.ylabel('MonthlyCharge - $')
# plt.title('6 Customer Clusters')
# plt.show()

#Tenure and Bandwidth
#Training the Method
kmeans = KMeans(n_clusters = 2, init = 'k-means++', random_state = 42)

y_kmeans = kmeans.fit_predict(X)
print('Y_Kmeans are: ',y_kmeans)

#Cluster Visualization
```

```
plt.scatter(X[y_kmeans == 0,0], X[y_kmeans == 0,1], s = 10, c = 'red', label
= 'Cluster #1')
plt.scatter(X[y_kmeans == 1,0], X[y_kmeans == 1,1], s = 10, c = 'blue', label
= 'Cluster #2')

#Plot the Centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s =
100, c = 'yellow', label = 'Centroids')
plt.xlabel('Tenure - months')
plt.ylabel('Bandwidth_GB_Year')
plt.title('2 Customer Clusters')
plt.show()
```

The clustering analysis is divided into three parts:

Part I – Analyzing Tenure and MonthylCharge
Part II – Analyzing Income and MonthlyCharge
Part III – Analyzing Tenure and Bandwidth_GB_Year

PART I: Tenure and MonthlyCharge

```
#Starting the KMeans Analysis
df = pd.read_csv('prepared_churn_data_kmeans.csv', index_col= 0)

#Importing KMeans
from sklearn.cluster import KMeans

#Selecting ggplot to make it look pretty
plt.style.use('ggplot')

#Selecting the features Tenure and MonthlyCharge (They are respectively
columns number 30 and 31)
X = churn_num.iloc[:,[2,1]].values

#Create a WCSS list
wcss = []

#Finding the optimal number of clusters
for i in range (1,10):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

#Plotting the optimal number
plt.plot(range(1,10), wcss)
plt.title('The Elbow Method Using Inertia')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```

The first step is to find the ideal number of clusters with the "elbow method" using inertia. Inertia is the sum of squared distances of samples to their closest cluster center[3].

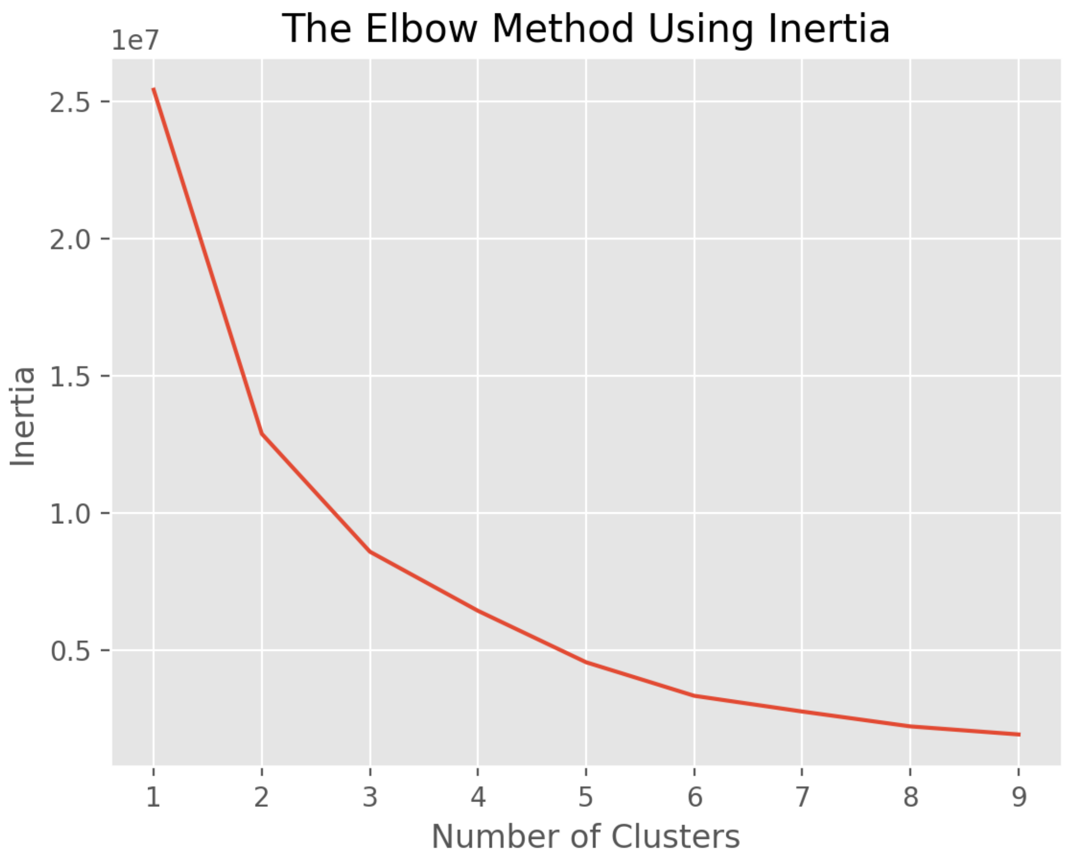


Figure 10: Finding the Optimal Number of Clusters

The optimal number in this case is 6. So I applied it in variable "n_clusters" to properly train the method:

```
# Tenure and Monthly Charge
#Training the Method
kmeans = KMeans(n_clusters = 6, init = 'k-means++', random_state = 42)

y_kmeans = kmeans.fit_predict(X)
print('Y_Kmeans are: ',y_kmeans)

#Cluster Visualization
plt.scatter(X[y_kmeans == 0,0], X[y_kmeans == 0,1], s = 10, c = 'red', label
= 'Cluster #1')
plt.scatter(X[y_kmeans == 1,0], X[y_kmeans == 1,1], s = 10, c = 'blue', label
= 'Cluster #2')
plt.scatter(X[y_kmeans == 2,0], X[y_kmeans == 2,1], s = 10, c = 'orange',
label = 'Cluster #3')
plt.scatter(X[y_kmeans == 3,0], X[y_kmeans == 3,1], s = 10, c = 'magenta',
label = 'Cluster #4')
plt.scatter(X[y_kmeans == 4,0], X[y_kmeans == 4,1], s = 10, c = 'green',
label = 'Cluster #5')
plt.scatter(X[y_kmeans == 5,0], X[y_kmeans == 5,1], s = 10, c = 'cyan', label
= 'Cluster #6')

#Plot the Centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s =
100, c = 'yellow', label = 'Centroids')
plt.xlabel('Tenure - months')
plt.ylabel('MonthlyCharge - $')
plt.title('6 Customer Clusters')
plt.show()
```

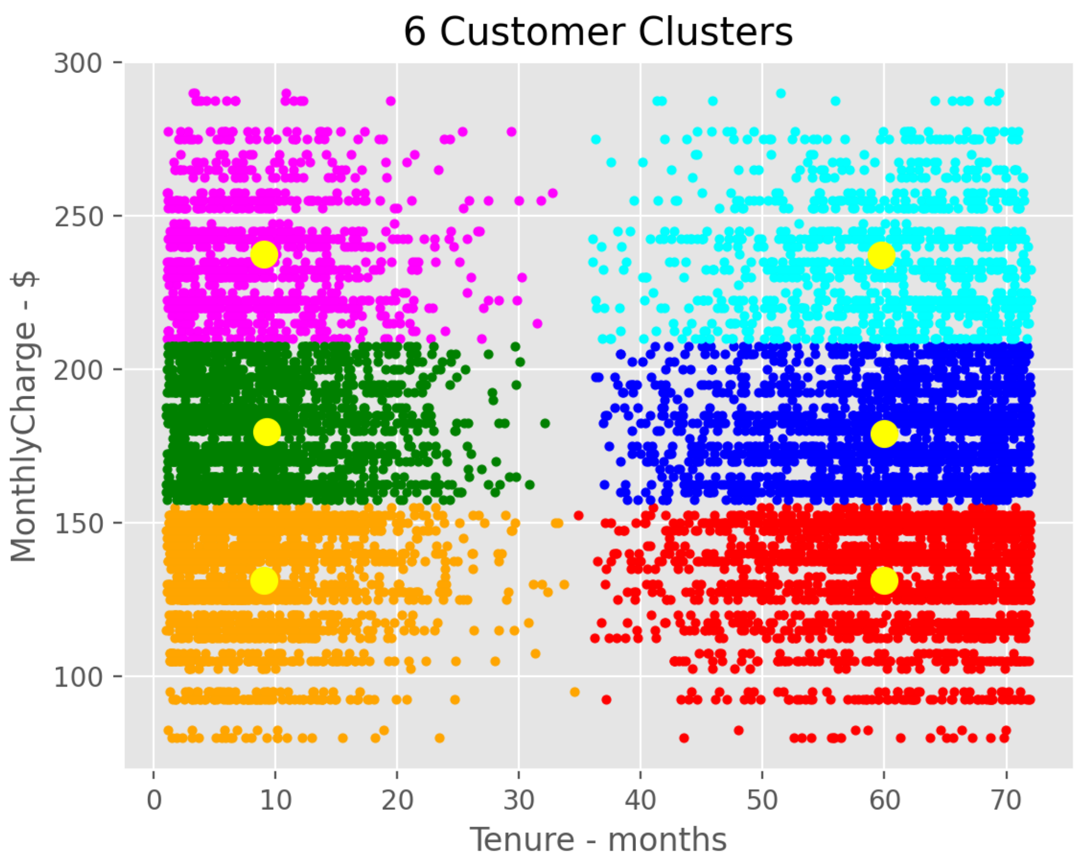


Figure 11: Tenure and MonthlyCharge Cluster

PART II – Income and MonthlyCharge

```
#Importing KMeans
from sklearn.cluster import KMeans

#Selecting ggplot to make it look pretty
plt.style.use('ggplot')
```

```
#Selecting the features Income and MonthlyCharge
X = df.iloc[:,[4,1]].values

#Create a WCSS list
wcss = []

#Finding the optimal number of clusters
for i in range (1,10):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

#Plotting the optimal number
plt.plot(range(1,10), wcss)
plt.title('The Elbow Method Using Inertia')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```

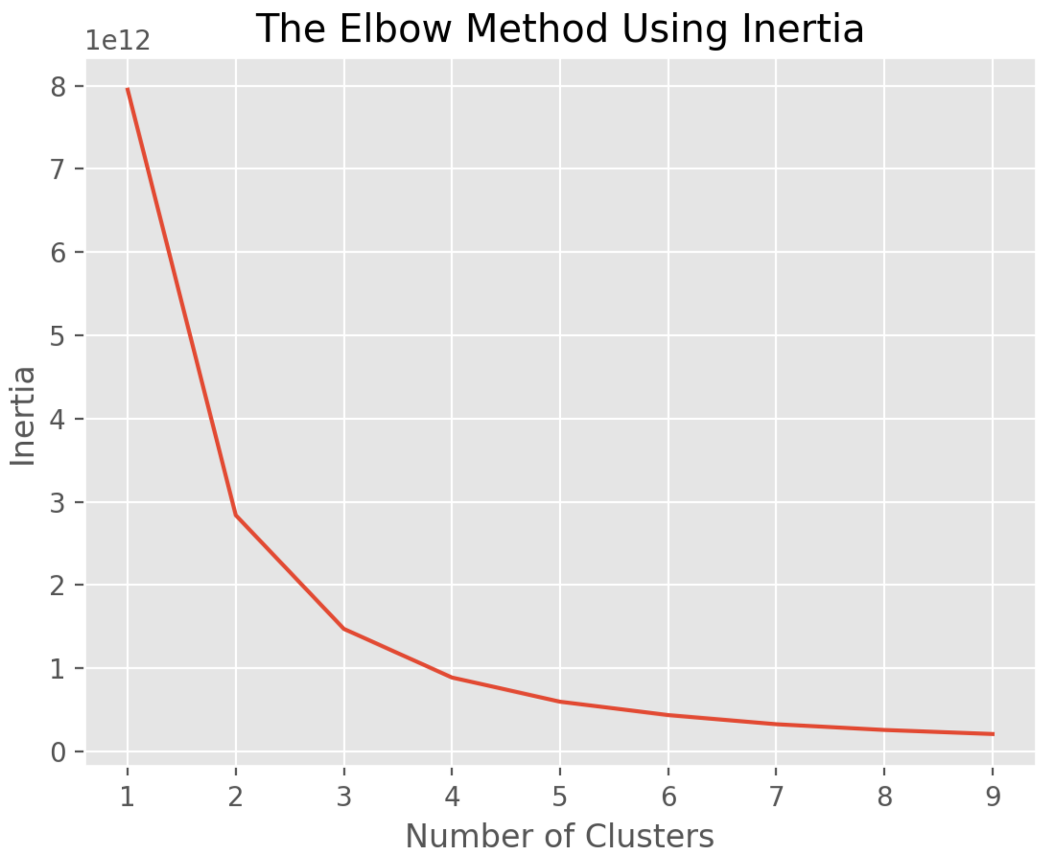


Figure 12: Finding the Optimal Number of Clusters

In this case the number of optimal clusters is 4.

```
#Income and Monthly Charge
#Training the Method
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)

y_kmeans = kmeans.fit_predict(X)
print('Y_Kmeans are: ',y_kmeans)

#Cluster Visualization
plt.scatter(X[y_kmeans == 0,0], X[y_kmeans == 0,1], s = 10, c = 'red', label
= 'Cluster #1')
plt.scatter(X[y_kmeans == 1,0], X[y_kmeans == 1,1], s = 10, c = 'blue', label
= 'Cluster #2')
plt.scatter(X[y_kmeans == 2,0], X[y_kmeans == 2,1], s = 10, c = 'orange',
label = 'Cluster #3')
plt.scatter(X[y_kmeans == 3,0], X[y_kmeans == 3,1], s = 10, c = 'magenta',
label = 'Cluster #4')

#Plot the Centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s =
100, c = 'yellow', label = 'Centroids')
plt.xlabel('Income - $')
plt.ylabel('MonthlyCharge - $')
plt.title('4 Customer Clusters')
plt.show()
```

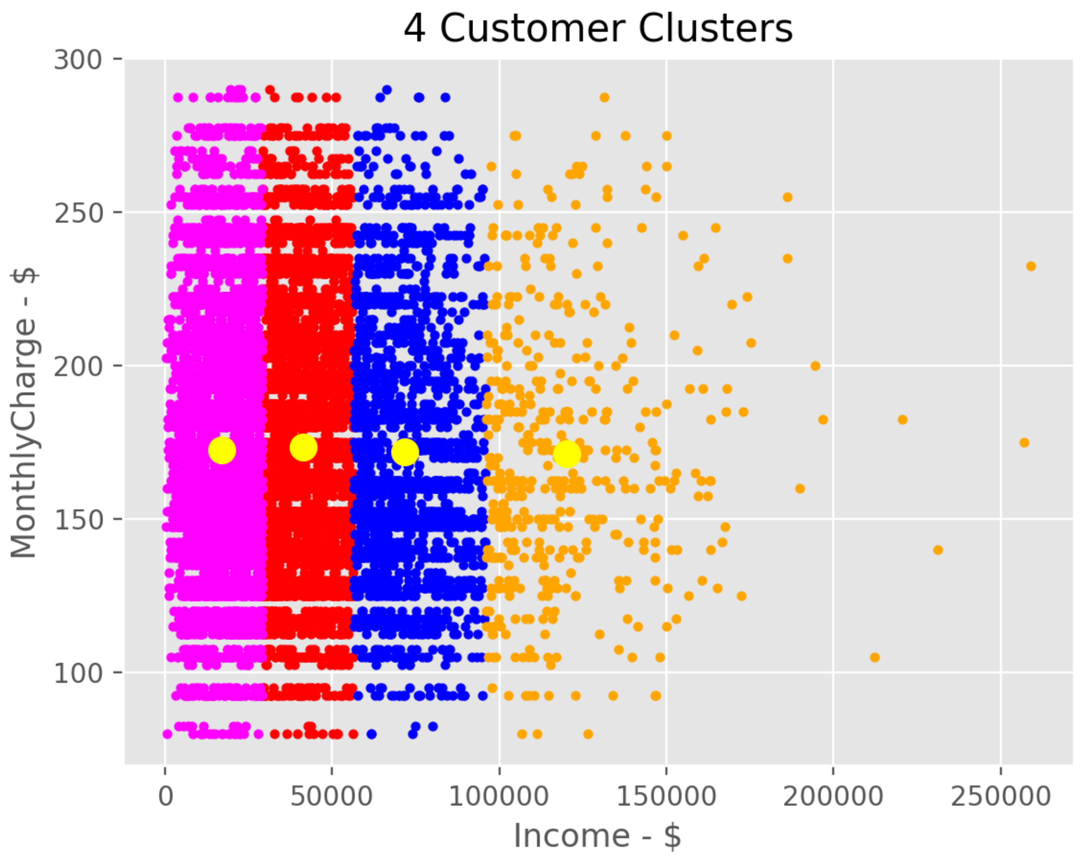


Figure 13: Income and MonthlyCharge Clusters

PART III – Tenure and Bandwidth_GB_Year

```
#Importing KMeans
from sklearn.cluster import KMeans

#Selecting ggplot to make it look pretty
plt.style.use('ggplot')
```

```
#Selecting the features Tenure and Bandwidth
X = df.iloc[:,[2,3]].values
```

```
#Create a WCSS list
wcss = []

#Finding the optimal number of clusters
for i in range (1,10):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

```
#Plotting the optimal number
plt.plot(range(1,10), wcss)
plt.title('The Elbow Method Using Inertia')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```

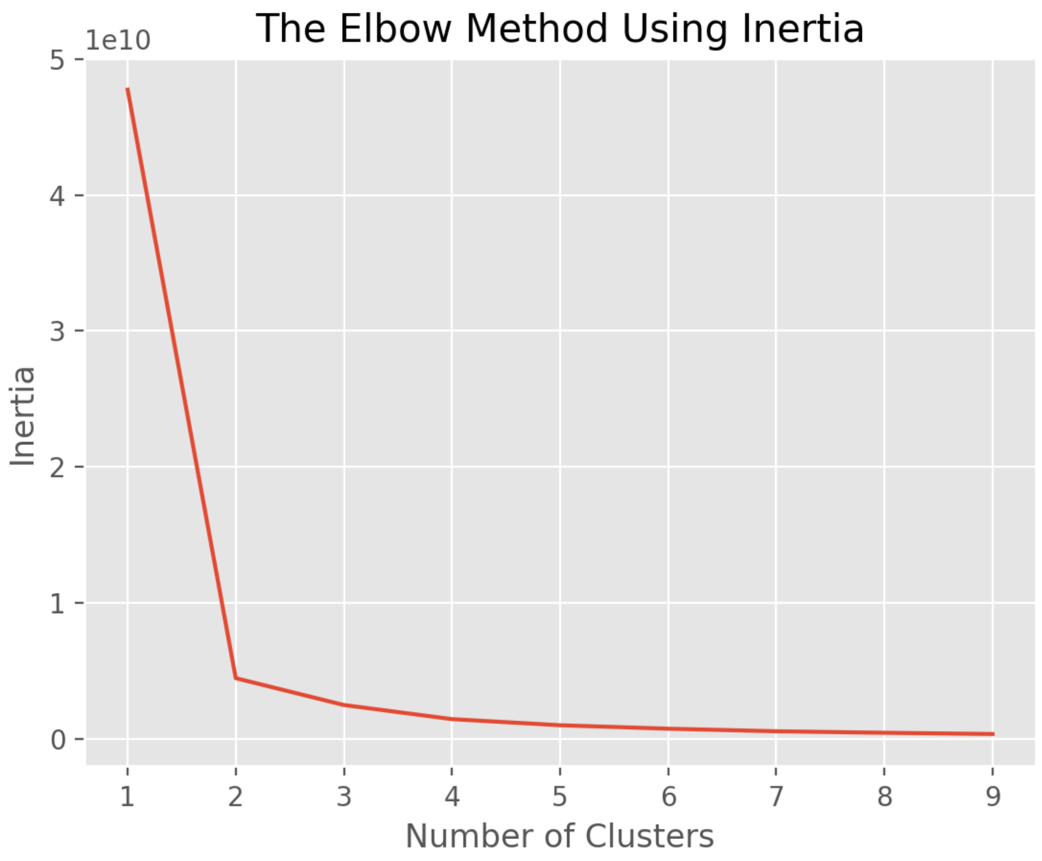


Figure 14: Finding the Optimal Number of Clusters

In this case, the optimal number of clusters is 2.

```
#Tenure and Bandwidth
#Training the Method
kmeans = KMeans(n_clusters = 2, init = 'k-means++', random_state = 42)

y_kmeans = kmeans.fit_predict(X)
print('Y_Kmeans are: ',y_kmeans)

#Cluster Visualization
plt.scatter(X[y_kmeans == 0,0], X[y_kmeans == 0,1], s = 10, c = 'red', label
= 'Cluster #1')
plt.scatter(X[y_kmeans == 1,0], X[y_kmeans == 1,1], s = 10, c = 'blue', label
= 'Cluster #2')
```

```
#Plot the Centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s =
100, c = 'yellow', label = 'Centroids')
plt.xlabel('Tenure - months')
plt.ylabel('Bandwidth_GB_Year')
plt.title('2 Customer Clusters')
plt.show()
```

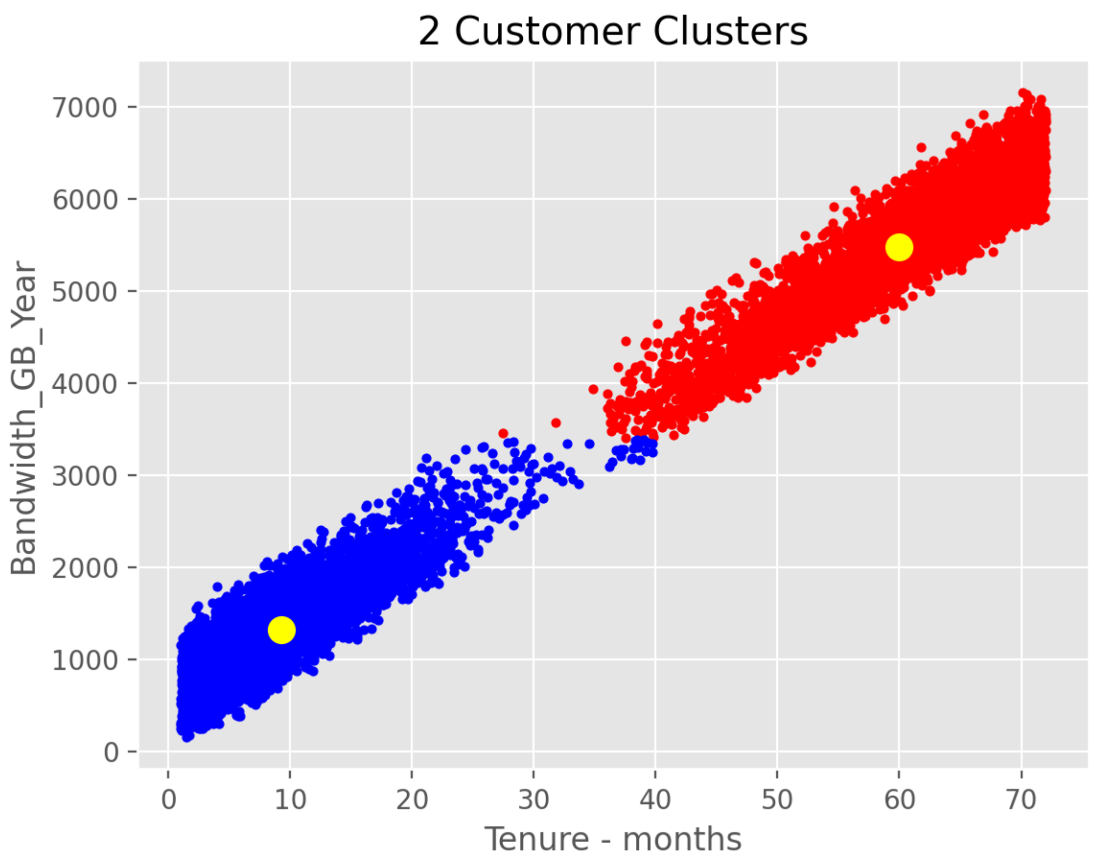


Figure 15: Tenure and Bandwidth Clusters

Part V: Data Summary and Implications

E. Summarize your data analysis by doing the following:

1. Explain the accuracy of your clustering technique.

K-means is not a classification tool, and providing an accuracy score is hard. Therefore, let's consider clustering quality and number of clusters as measures of accuracy.

Number of Clusters: We have used a visual tool to find the optimal number of clusters for each scenario: the elbow method. Visualization is not accurate, and this can negatively add to the accuracy of the clustering technique.

Quality: Our points are not tightly gathered around the centroids which could mean the method has low accuracy.

Based on the two factors explained above, it's ok to assume that this clustering method does not present a high level of accuracy.

2. Discuss the results and implications of your clustering analysis.

Through analysis of the monthly charge and tenure (Figure 11), we see that one of the customer groups retained for the longest time, also has a lower monthly payment (red cluster). The majority of customers are paying below $200 a month.

In Figure 13, we can see that there are a considerable number of customers in the low income area who also have high monthly payments. Since these customers are not able to afford the higher price, the natural consequence is to lose them.

Figure 15 illustrates very clearly that customers who use a lot of bandwidth each year tend to stay longer with the company. Ergo, it is recommended that the company offer a wider variety of options during the customer onboarding process. The more bandwidth per year used, the more likely that customer will be retained for a longer period of time.

3. Discuss **one** limitation of your data analysis.

The data analysis limitation is the lack of ability to further question the data acquisition. During a real-world project, talking to the department in charge of acquiring the data would help to understand a few characteristics of the data and would allow discussions about the data collection methodology, etc.

4. Recommend a course of action for the real-world organizational situation from part A1 based on your results and implications discussed in part E2.

Stakeholders and the marketing department need to work closely to come up with specific service offerings for both low and high income customers. The company will continue to see high churn rates if low income customers keep signing up for services they cannot afford.

We know that the more bandwidth a customer uses per year, the higher the chance is that customer will be retained. Therefore, the key to increasing customer retention is to maximize the amount of services that each customer is offered, based on what they are able to pay.

Part VI: Demonstration

F. Provide a Panopto video recording that includes a demonstration of the functionality of the code used for the analysis and a summary of the programming environment.

Online Path for Panopto video recording:
https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=491af779-7ff9-41fe-adf1-ae030188c40a

H. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

[1] (2018, Sept 12th) GARBADE, Dr. Michael J., Understanding K-Means Clustering in Machine Learning, https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[2] (2015, Jan 16th) K-means clustering is not a free lunch, http://varianceexplained.org/r/kmeans-free-lunch/

[3] (2021, Feb 9th) GUPTA, Alind, Elbow Method for Optimal Value of k in KMeans https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

[4] (2020, Jun 10th) KUMAR, Satyam, Understanding K-means, K-means++, and K-Medoids Clustering Algorithms https://towardsdatascience.com/understanding-k-means-k-means-and-k-medoids-clustering-algorithms-ad9c9fbf47ca