D212 – DATA MINING II

Task 1: Clustering Techniques

WGU - MSDA

Advanced Data Mining using K-Means Clustering Technique and Churn Dataset

Richard Flores
Rflo147@wgu.edu
Student ID: 006771163

Table of Contents

Part I: Research Question

A1. Proposal of Question

As the telecommunications market becomes increasingly competitive with new and improved technologies including free applications like META (Facebook) messenger, Telegram, and TikTok the need for customer retention is becoming critically important.

The question answered in this research project is:

How do we identify customers at risk of churn and what telecom services or features are correlated?

I will be using the K-Means Clustering Technique in this Data Mining analysis.

A2. Defined Goal

The goal of the research question is to provide stakeholders direct and actionable insight to create a plan for operations personnel, officers, and managers to increase customer satisfaction through targeted services observed in the dataset and to reduce customer churn and protect long-term profits.

Part II: Technique Justification

B1. Explanation of Clustering Technique

In this Data Mining analysis, I have chosen to use the K-Means clustering technique as it is one of the most popular and efficient unsupervised algorithms available.

As AndreyBu from TowardsDataScience explains, “the objective of K-means is simple: group similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number (*k*) of clusters in a dataset” (Garbade 2018).

The K-Means algorithm works by first selecting a group of random centroids, this serves as the beginning point for each additional cluster. The algorithm then executes iterative calculations to efficiently optimize the centroids positions. The algorithm has successfully completed its operations when one of two conditions are met. The first condition is stabilization of the centroids or when there are no further changes to their values. The second condition is achieved with the predetermined number of iterations has been completed.

From a data mining analysis of the Churn dataset using k-means clustering, we expect to see illustrations of clusters that display similarities and differences between groups of customer types and defined market segments.

The outcome of the data mining analysis should observe customer churn depending on various features in the dataset such as tenure, a pattern in subscribed services, or customer satisfaction based on customer service experiences.

B2. Summary of Technique Assumption

The assumption of the k-means clustering algorithm is that datapoints will cluster together based on calculated similarities. In this analysis we will define K, which points to the number of centroids necessary in the dataset. A centroid is defined as an imagined or real location representation of the center of the cluster (sklearn 2022).

By reducing the in-cluster sum of squares, every data point is allocated to each of the clusters. The k-means data mining algorithm identifies the number of centroids and allocates each one to the nearest cluster while minimizing the number of centroids available.

B3. Packages or Libraries List

For this Data Mining analysis, I will be using the Python language and the following packages or libraries:

Data Science Libraries
- NumPy
- Pandas

Visualization Libraries
- Seaborn
- Matplotlib

Predictive Analysis
- Scikit-Learn

<u>Justification for libraries and packages in support of the Data Mining Analysis</u>

NumPy – NumPy is integral for performing mathematical and logical operations on arrays. It provides many of the functions needed to manipulate n-arrays and matrices in Python. This includes how to create NumPy arrays, broadcasting, accessing values, and managing arrays.

Pandas – Pandas is used to infer and analyze data in Python. Pandas is used for data cleanup, transformation, management and analysis of the cleaned churn dataset.

Seaborn – Seaborn takes each data frame or array that contains information and performs internal functions necessary to integrate semantic mapping and statistics to turn the data into visual representations.

Matplotlib – Matplotlib is a plotting library for creating 2D plots in Python. It consists of a set of graphing plots such as line plots, bar plots, frequency distribution plots, and histograms and can display different types of data.

Scikit-Learn - Scikit-learn is a library that provides many supervised and unsupervised learning algorithms in Python. Functions provided by Scikit-learn include Regression, linear and logistic regression as well as classification including K-Nearest Neighbors.

Part III: Data Preparation

C1. Data Preprocessing

As with the previous Multiple and Logistic regression analysis, a preprocessing data goal is to convert binary responses in the dataset i.e. 'Yes' or 'No' into dummy variables using numerical '1' or '0' variables in order to enable statistical analysis.

For example, converting customer responses if they have "TechSupport" from 'No' to '0' and changing 'Yes' to '1'.

C2. Data Set Variables

This analysis will use the following 9 continuous variables and 13 categorical variables.

Continuous variables include:

- Bandwidth_GB_Year
- Children
- Contacts
- Email
- Income
- MonthlyCharge
- Outage_sec_perweek
- Tenure
- Yearly_equip_failure

Categorical variables include:

- Contract
- DeviceProtection
- InternetService
- Multiple
- OnlineBackup
- OnlineSecurity
- Phone
- Port_modem
- StreamingMovies
- StreamingTV
- Tablet
- TechSupport
- Techie

In addition, the customer survey responses represent ordinal predictors, listed as follows:

Item1 - Timely response
Item2 - Timely fixes
Item3 - Timely replacements
Item4 - Reliability
Item5 - Options
Item6 - Respectful Response
Item7 - Courteous Exchange
Item8 – Evidence of Active Listening

C3. Steps for Analysis

▪ Import the 'clean_churn' dataset into a Pandas dataframe for analysis.
▪ Rename features in the survey responses to better describe the items.
▪ Describe the various features and data to prepare relevant items.
▪ Create a view of the summary statistics.
▪ After review, remove features that are not relevant to analyzing the target variable.
▪ Review record data to check for anomalies, outliers, missing data and other data that could become obstacles in the analysis.
▪ Utilize dummy variables in order to numerically analyze data by changing "Yes/No" responses to binary "1/0" responses.
▪ Export manipulated Dataframe to .CSV for analysis in k-means data mining model.

```
# Standard library imports, and Visualization, Statistics, SciKit libraries
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import sklearn
from sklearn import datasets
from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# Ignore Warning messages
import warnings
warnings.filterwarnings('ignore')

import matplotlib as mpl
COLOR = 'white'
mpl.rcParams['text.color'] = COLOR
mpl.rcParams['axes.labelcolor'] = COLOR
mpl.rcParams['xtick.color'] = COLOR
mpl.rcParams['ytick.color'] = COLOR
```

```
# Load churn dataset into a Pandas dataframe
churn_df = pd.read_csv('churn_clean.csv', index_col=0)
```

```
# List columns in the dataframe
churn_df.columns
```

```
Index(['Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip',
       'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Children',
       'Age', 'Income', 'Marital', 'Gender', 'Churn', 'Outage_sec_perweek',
       'Email', 'Contacts', 'Yearly_equip_failure', 'Techie', 'Contract',
       'Port_modem', 'Tablet', 'InternetService', 'Phone', 'Multiple',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'PaymentMethod',
       'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Item1', 'Item2',
       'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

```
# Verify the number of records and columns in the dataset
churn_df.shape
```

```
(10000, 49)
```

(DataCamp 2021)

```python
# Verify headers of imported dataset
churn_df.head()
```

CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Population	...	MonthlyChar
1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	38	...	172.455
2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	10446	...	242.632
3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	...	159.947
4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.24798	13863	...	119.956
5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	...	149.948

5 rows × 49 columns

```python
# Verify dataset info
churn_df.info
```

```
<bound method DataFrame.info of           Customer_id                           Interaction  \
CaseOrder
1             K409198  aa90260b-4141-4a24-8e36-b04ce1f4f77b
2             S120509  fb76459f-c047-4a9d-8af9-e0f7d4ac2524
3             K191035  344d114c-3736-4be5-98f7-c72c281e2d35
4              D90850  abfa2b40-2d43-4994-b15a-989b8c79e311
5             K662701  68a861fd-0d20-4e51-a587-8a90407ee574
...               ...                                   ...
9996          M324793  45deb5a2-ae04-4518-bf0b-c82db8dbe4a4
9997          D861732  6e96b921-0c09-4993-bbda-a1ac6411061a
9998          I243405  e8307ddf-9a01-4fff-bc59-4742e03fd24f
9999          I641617  3775ccfc-0052-4107-81ae-9657f81ecdf3
10000          T38070  9de5fb6e-bd33-4995-aec8-f01d0172a499

                                        UID         City State  \
CaseOrder
1          e885b299883d4f9fb18e39c75155d990   Point Baker    AK
2          f2de8bef964785f41a2959829830fb8a   West Branch    MI
3          f1784cfa9f6d92ae816197eb175d3c71       Yamhill    OR
4          dc8a365077241bb5cd5ccd305136b05e       Del Mar    CA
5          aabb64a116e83fdc4befc1fbab1663f9     Needville    TX
...                                     ...          ...   ...
9996       9499fb4de537af195d16d046b79fd20a   Mount Holly    VT
9997       c09a841117fa81b5c8e19afec2760104   Clarksville    TN
9998       9c41f212d1e04dca84445019bbc9b41c      Mobeetie    TX
9999       3e1f269b40c235a1038863ecf6b7a0df    Carrollton    GA
10000      0ea683a03a3cd544aefe8388aab16176  Clarkesville    GA

                          County    Zip       Lat        Lng  Population  ...  \
CaseOrder                                                                ...
1          Prince of Wales-Hyder  99927  56.25100 -133.37571          38  ...
2                         Ogemaw  48661  44.32893  -84.24080       10446  ...
3                        Yamhill  97148  45.35589 -123.24657        3735  ...
4                      San Diego  92014  32.96687 -117.24798       13863  ...
5                      Fort Bend  77461  29.38012  -95.80673       11352  ...
...                          ...    ...       ...        ...         ...  ...
9996                     Rutland   5758  43.43391  -72.78734         640  ...
9997                  Montgomery  37042  36.56907  -87.41694       77168  ...
9998                     Wheeler  79061  35.52039 -100.44180         406  ...
9999                     Carroll  30117  33.58016  -85.13241       35575  ...
10000                  Habersham  30523  34.70783  -83.53648       12230  ...
```

(DataCamp 2021)

```
           MonthlyCharge Bandwidth_GB_Year Item1  Item2  Item3  Item4 Item5  \
CaseOrder
1             172.455519        904.536110     5      5      5      3     4
2             242.632554        800.982766     3      4      3      3     4
3             159.947583       2054.706961     4      4      2      4     4
4             119.956840       2164.579412     4      4      4      2     5
5             149.948316        271.493436     4      4      4      3     4
...                  ...               ...   ...    ...    ...    ...   ...
9996          159.979400       6511.252601     3      2      3      3     4
9997          207.481100       5695.951810     4      5      5      4     4
9998          169.974100       4159.305799     4      4      4      4     4
9999          252.624000       6468.456752     4      4      6      4     3
10000         217.484000       5857.586167     2      2      3      3     3

           Item6 Item7  Item8
CaseOrder
1              4     3      4
2              3     4      4
3              3     3      3
4              4     3      3
5              4     4      5
...          ...   ...    ...
9996           3     2      3
9997           5     2      5
9998           4     4      5
9999           3     5      4
10000          3     4      1

[10000 rows x 49 columns]>
```

```
# Describe Churn dataset
churn_df.describe()
```

	Zip	Lat	Lng	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	..
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	..
mean	49153.319600	38.757567	-90.782536	9756.562400	2.0877	53.078400	39806.926771	10.001848	12.016000	0.994200	..
std	27532.196108	5.437389	15.156142	14432.698671	2.1472	20.698882	28199.916702	2.976019	3.025898	0.988466	..
min	601.000000	17.966120	-171.688150	0.000000	0.0000	18.000000	348.670000	0.099747	1.000000	0.000000	..
25%	26292.500000	35.341828	-97.082812	738.000000	0.0000	35.000000	19224.717500	8.018214	10.000000	0.000000	..
50%	48869.500000	39.395800	-87.918800	2910.500000	1.0000	53.000000	33170.605000	10.018560	12.000000	1.000000	..
75%	71866.500000	42.106908	-80.088745	13168.000000	3.0000	71.000000	53246.170000	11.969485	14.000000	2.000000	..
max	99929.000000	70.640660	-65.667850	111850.000000	10.0000	89.000000	258900.700000	21.207230	23.000000	7.000000	..

8 rows × 22 columns

```
# List features available in the dataset
churn_df.dtypes
```

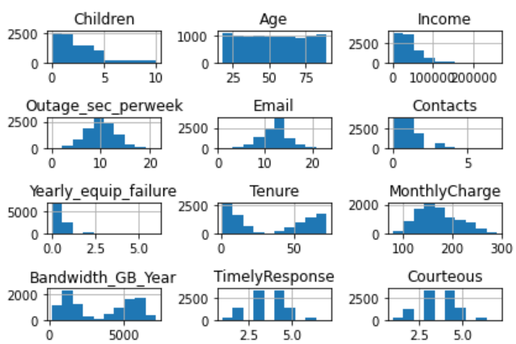
```
Customer_id           object
Interaction           object
UID                   object
City                  object
State                 object
County                object
Zip                    int64
Lat                  float64
Lng                  float64
Population             int64
Area                  object
TimeZone              object
Job                   object
Children               int64
Age                    int64
Income               float64
Marital               object
Gender                object
Churn                 object
```

(DataCamp 2021)

```
Outage_sec_perweek     float64
Email                    int64
Contacts                 int64
Yearly_equip_failure     int64
Techie                  object
Contract                object
Port_modem              object
Tablet                  object
InternetService         object
Phone                   object
Multiple                object
OnlineSecurity          object
OnlineBackup            object
DeviceProtection        object
TechSupport             object
StreamingTV             object
StreamingMovies         object
PaperlessBilling        object
PaymentMethod           object
Tenure                 float64
MonthlyCharge          float64
Bandwidth_GB_Year      float64
Item1                    int64
Item2                    int64
Item3                    int64
Item4                    int64
Item5                    int64
Item6                    int64
Item7                    int64
Item8                    int64
dtype: object
```

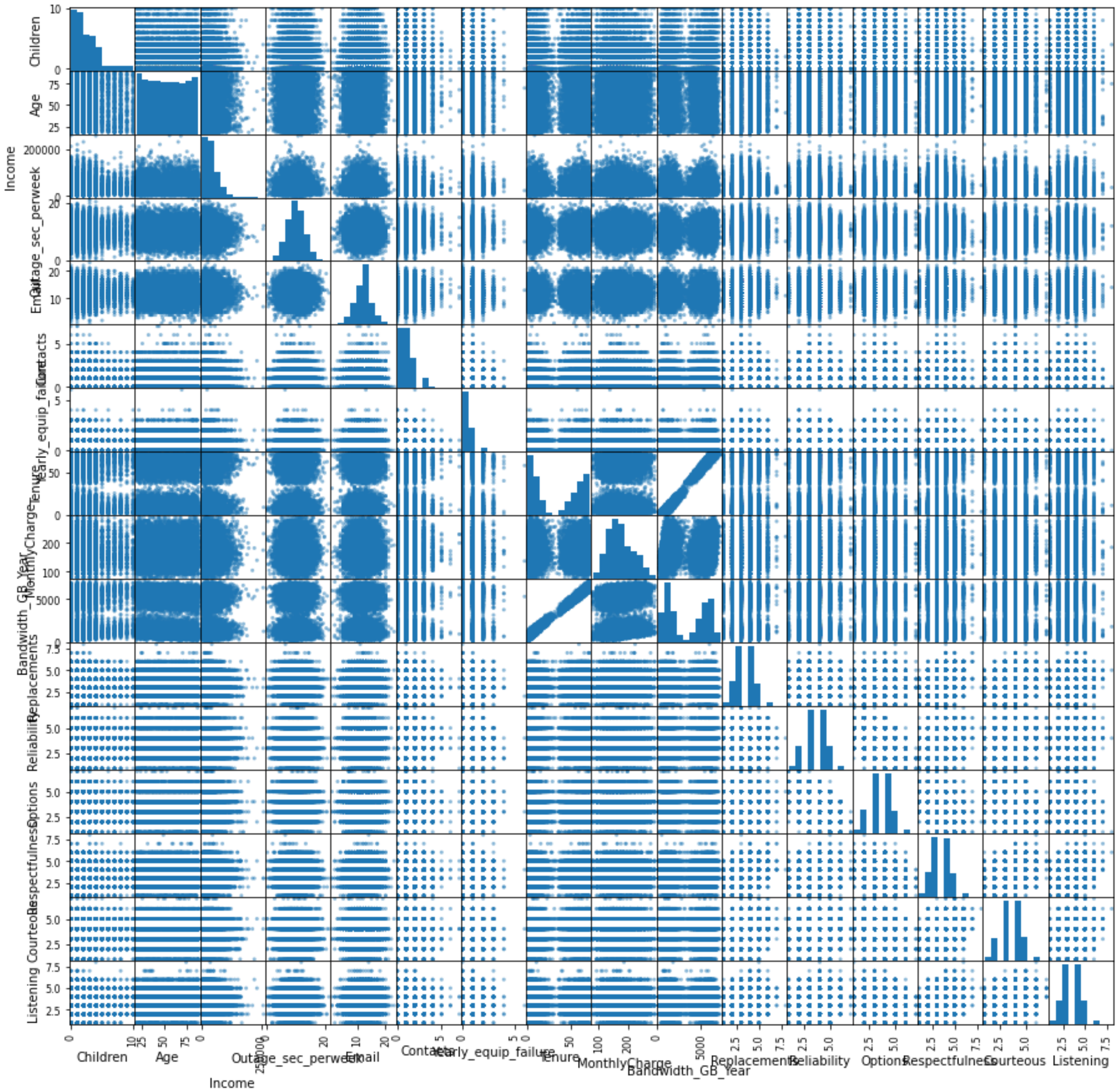
```
# Rename 8 customer survey features to represent descriptions for clarity
churn_df.rename(columns = {'Item1':'TimelyResponse',
                    'Item2':'Fixes',
                     'Item3':'Replacements',
                     'Item4':'Reliability',
                     'Item5':'Options',
                     'Item6':'Respectfulness',
                     'Item7':'Courteous',
                     'Item8':'Listening'},
          inplace=True)
```

```
# Display histograms of continuous & categorical variables
churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure',
          'MonthlyCharge', 'Bandwidth_GB_Year', 'TimelyResponse', 'Courteous']].hist()
plt.savefig('classification_pyplot.jpg')
plt.tight_layout()
```



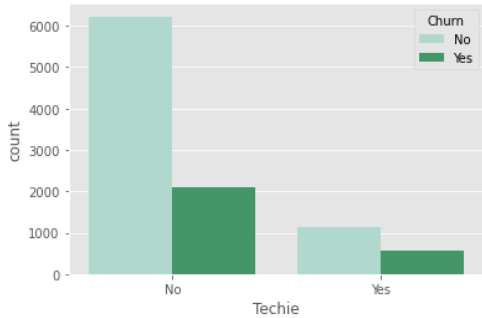
(DataCamp 2021)

```python
# Scatter matrixes of numeric variables for a broad overview of possible relationships.
churn_numeric = churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek',
                          'Email', 'Contacts','Yearly_equip_failure', 'Tenure',
                          'MonthlyCharge', 'Bandwidth_GB_Year', 'Replacements',
                          'Reliability', 'Options', 'Respectfulness', 'Courteous',
                          'Listening']]

pd.plotting.scatter_matrix(churn_numeric, figsize = [15, 15]);
```

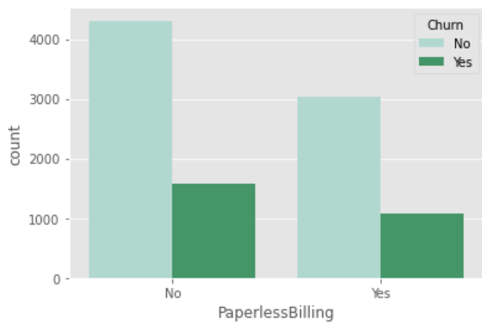


(DataCamp 2021)

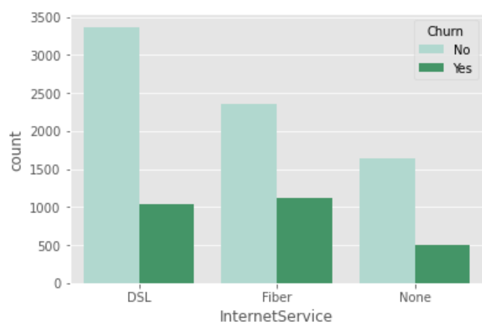
```python
# Enable ggplot
plt.style.use('ggplot')

# Countplot to show relationship of binary feature techie and churn
plt.figure()
sns.countplot(x='Techie', hue='Churn', data=churn_df, palette='BuGn')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```




```python
# Countplot to show relationship of binary feature PaperlessBilling and churn
plt.figure()
sns.countplot(x='PaperlessBilling', hue='Churn', data=churn_df, palette='BuGn')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```




```python
# Countplot to show relationship of binary feature InternetService and churn
plt.figure()
sns.countplot(x='InternetService', hue='Churn', data=churn_df, palette='BuGn')
plt.xticks([0,1,2], ['DSL', 'Fiber', 'None'])
plt.show()
```



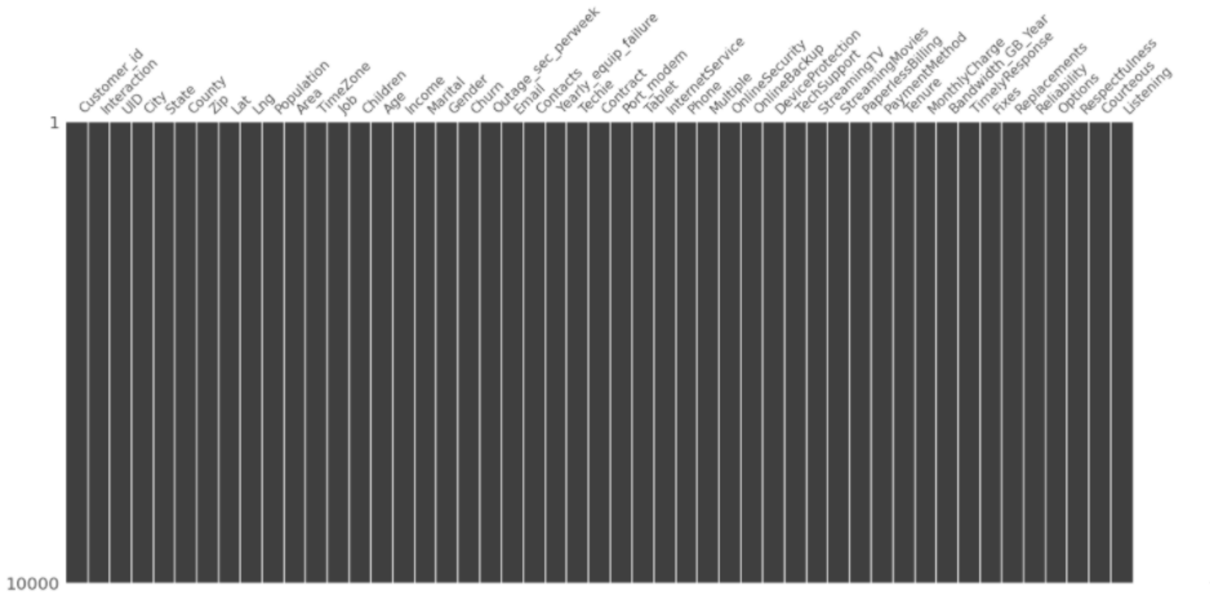
(DataCamp 2021)

```python
# Verify missing data points
data_nulls = churn_df.isnull().sum()
print(data_nulls)
```

```
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
TimelyResponse          0
Fixes                   0
Replacements            0
Reliability             0
Options                 0
Respectfulness          0
Courteous               0
Listening               0
dtype: int64
```

(DataCamp 2021)

```python
# Visualize missing values in dataset using missingno

!pip install missingno
import missingno as msno

# Display matrix to visualize any missing values
msno.matrix(churn_df);
```




```python
# Convert all "Yes/No" data into binary "1/0" representation
churn_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in churn_df['Churn']]
churn_df['DummyContract'] = [1 if v == 'Two Year' else 0 for v in churn_df['Contract']]
churn_df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in churn_df['DeviceProtection']]
churn_df['DummyGender'] = [1 if v == 'Male' else 0 for v in churn_df['Gender']]
churn_df['DummyInternetService'] = [1 if v == 'Fiber Optic' else 0 for v in churn_df['InternetService']]
churn_df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in churn_df['Multiple']]
churn_df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in churn_df['OnlineBackup']]
churn_df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in churn_df['OnlineSecurity']]
churn_df['DummyPaperlessBilling'] = [1 if v == 'Yes' else 0 for v in churn_df['PaperlessBilling']]
churn_df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in churn_df['Phone']]
churn_df['DummyPort_modem'] = [1 if v == 'Yes' else 0 for v in churn_df['Port_modem']]
churn_df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in churn_df['StreamingTV']]
churn_df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in churn_df['Tablet']]
churn_df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in churn_df['TechSupport']]
churn_df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in churn_df['Techie']]
churn_df['StreamingMovies'] = [1 if v == 'Yes' else 0 for v in churn_df['StreamingMovies']]
```

```python
# Remove redundant 'yes/no' features from dataframe
churn_df = churn_df.drop(columns=['Gender', 'Churn', 'Techie', 'Contract', 'Port_modem', 'Tablet',
                                  'InternetService', 'Phone', 'Multiple', 'OnlineSecurity',
                                  'OnlineBackup', 'DeviceProtection', 'TechSupport',
                                  'StreamingTV', 'StreamingMovies', 'PaperlessBilling'])
```

(DataCamp 2021)

```python
# Remove features not relevant to the proposed analysis question
churn_df = churn_df.drop(columns=['Customer_id', 'Interaction', 'UID',
                         'City', 'State', 'County', 'Zip', 'Lat', 'Lng',
                         'Area', 'TimeZone', 'Job', 'Marital', 'PaymentMethod'])
churn_df.head()
```

	Population	Children	Age	Income	Outage_sec_perweek	Email	Contacts	Yearly_equip_failure	Tenure	MonthlyCharge	...	DummyMultiple	Du
CaseOrder													
1	38	0	68	28561.99	7.978323	10	0	1	6.795513	172.455519	...	0	
2	10446	1	27	21704.77	11.699080	12	0	1	1.156681	242.632554	...	1	
3	3735	4	50	9609.57	10.752800	9	0	1	15.754144	159.947583	...	1	
4	13863	1	48	18925.23	14.913540	15	2	0	17.087227	119.956840	...	0	
5	11352	0	83	40074.19	8.147417	16	2	1	1.670972	149.948316	...	0	

5 rows × 34 columns

```python
# Display features in churn dataframe
features = (list(churn_df.columns[:-1]))
print('Analysis Features: \n', features)
```

```
Analysis Features:
 ['CaseOrder', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'Month
lyCharge', 'Bandwidth_GB_Year', 'TimelyResponse', 'Fixes', 'Replacements', 'Reliability', 'Options', 'Respectfulness', 'Courteo
us', 'Listening', 'DummyGender', 'DummyTechie', 'DummyContract', 'DummyPort_modem', 'DummyTablet', 'DummyInternetService', 'Dum
myPhone', 'DummyMultiple', 'DummyOnlineSecurity', 'DummyOnlineBackup', 'DummyDeviceProtection', 'DummyTechSupport', 'DummyStrea
mingTV', 'DummyPaperlessBilling']
```

(DataCamp 2021)

C4. Cleaned Data Set

```python
# Extract cleaned dataset to CSV
churn_df.to_csv('kmeans_prepared.csv')
```

Part IV: Analysis

D1. Output and Intermediate Calculations

The primary library used for intermediate calculations will be sklearn from scikit-learn. From sklearn we will import the KMeans module. The official scikit-learn documentation states that k-means uses Lloyd's or Elkan's algorithm for calculations (Sklearn 2022).

Sklearn calculates average complexity by using the expression O(k n T) where n represents the number of samples and T represents the amount of iteration (Sklearn 2022).

The k-means algorithm is one of the most efficient clustering algorithms available, but the algorithm falls in local minima.

The output of the intermediate calculations, or optimal clusters calculated using the elbow method, is displayed below in the code in the following section 'Code Execution'.

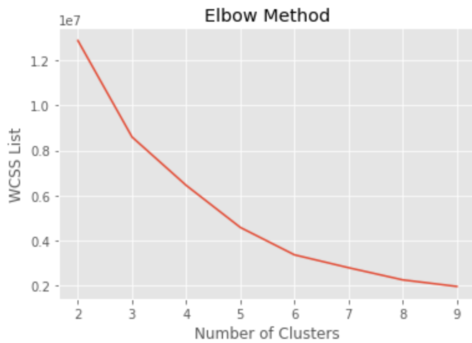
D2. Code Execution

```
# Standard library imports, and Visualization, Statistics, SciKit libraries
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import sklearn
from sklearn import datasets
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report

# SciPy
from scipy.cluster.vq import kmeans, vq

# Matplotlib
import matplotlib as mpl

# Import KMeans class from Scikit-Learn
from sklearn.cluster import KMeans

# Set plot style to ggplot for aesthetics & R style
plt.style.use('ggplot')
```

```
# Load churn dataset into a Pandas dataframe
churn_df = pd.read_csv('kmeans_prepared.csv', index_col=0)
```

```
# Create initial cluster indexes for Tenure and MonthlyCharge
X = churn_df.iloc[:, [35, 36]].values
```

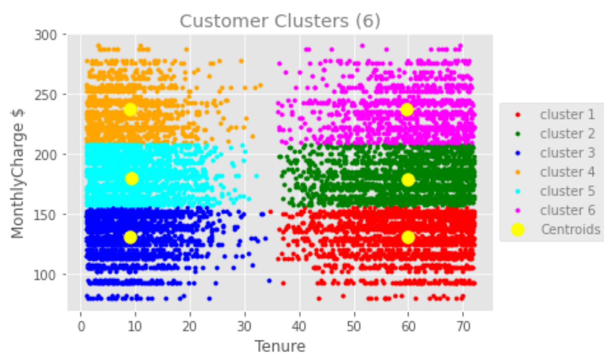
```python
# Calculate optimal amount of clusters with the elbow method
# Instantiate a (WCSS) Within Cluster Sum of Squares list
wcss = []

# Utilize a for loop to append values to WCSS list by iterating through kmeans datapoints
for i in range(2, 10):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Scree plot showing the optimal number of clusters
plt.plot(range(2, 10), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS List')
plt.savefig('tenure_vs_monthlycharge_plot1.jpg')
plt.show()
```




```python
# Train the K-means model
kmeans = KMeans(n_clusters=6, init='k-means++', random_state=42)

# Dependent variable utilized to split customers into different clusters
y_kmeans = kmeans.fit_predict(X)
```

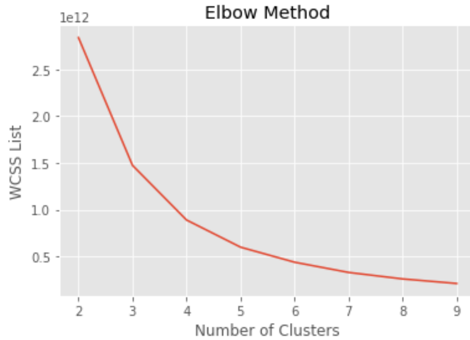
```python
# Create Scatter plot using 5 clusters for Tenure v. MonthlyCharge
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 10, c = 'red', label = 'cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 10, c = 'green', label = 'cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 10, c = 'blue', label = 'cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 10, c = 'orange', label = 'cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 10, c = 'cyan', label = 'cluster 5')
plt.scatter(X[y_kmeans == 5, 0], X[y_kmeans == 5, 1], s = 10, c = 'magenta', label = 'cluster 6')

# Plot centroids for each cluster
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'yellow', label = 'Centroids')

# Plot Labels
title_obj = plt.title('Customer Clusters (6)')
plt.getp(title_obj)
plt.getp(title_obj, 'text')
plt.setp(title_obj, color='gray')

plt.xlabel('Tenure')
plt.ylabel('MonthlyCharge $')

legend = plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.setp(legend.get_texts(), color='gray')

# Save plot
plt.savefig('tenure_vs_monthlycharge_plot2.jpg')

# Display Plot
plt.show();
```

```
agg_filter = None
alpha = None
animated = False
bbox_patch = None
children = []
clip_box = None
clip_on = True
clip_path = None
color or c = black
contains = None
figure = Figure(432x288)
fontfamily or family = ['sans-serif']
fontname or name = DejaVu Sans
fontproperties or font or font_properties = sans\-serif:style=normal:variant=normal:weight=nor...
fontsize or size = 14.399999999999999
fontstyle or style = normal
fontvariant or variant = normal
fontweight or weight = normal
gid = None
horizontalalignment or ha = center
in_layout = True
label =
path_effects = []
picker = None
position = (0.5, 1.0)
rasterized = None
rotation = 0.0
rotation_mode = None
sketch_params = None
snap = None
stretch = normal
text = Customer Clusters (6)
transform = CompositeGenericTransform(     BboxTransformTo(    ...
transformed_clip_path_and_affine = (None, None)
unitless_position = (0.5, 1.0)
url = None
usetex = False
verticalalignment or va = baseline
visible = True
wrap = False
zorder = 3
```




```
# Create initial cluster indexes for Income and MonthlyCharge
X = churn_df.iloc[:, [12, 36]].values
```

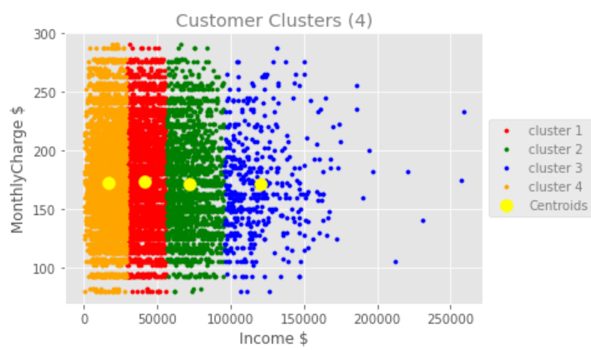
```
# Calculate optimal amount of clusters with the elbow method
# Instantiate a (WCSS) Within Cluster Sum of Squares list
wcss = []

# Utilize a for loop to append values to WCSS list by iterating through kmeans datapoints
for i in range(2, 10):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Scree plot showing the optimal number of clusters
plt.plot(range(2, 10), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS List')
plt.savefig('income_v_monthlycharge_plot1.jpg')
plt.show()
```




```python
# Train the K-means model
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)

# Dependent variable utilized to split customers into different clusters
y_kmeans = kmeans.fit_predict(X)
```

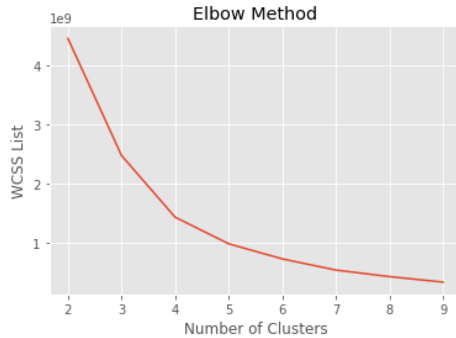
```python
# Create Scatter plot using 5 clusters for Income v. MonthlyCharge
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 10, c = 'red', label = 'cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 10, c = 'green', label = 'cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 10, c = 'blue', label = 'cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 10, c = 'orange', label = 'cluster 4')

# Plot centroids for each cluster
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'yellow', label = 'Centroids')

# Plot Labels
title_obj = plt.title('Customer Clusters (4)')
plt.getp(title_obj)
plt.getp(title_obj, 'text')
plt.setp(title_obj, color='gray')

plt.xlabel('Income $')
plt.ylabel('MonthlyCharge $')

legend = plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.setp(legend.get_texts(), color='gray')

# Save plot
plt.savefig('income_v_monthlycharge_plot2.jpg')

# Display Plot
plt.show();
```

```
agg_filter = None
alpha = None
animated = False
bbox_patch = None
children = []
clip_box = None
clip_on = True
clip_path = None
color or c = black
contains = None
figure = Figure(432x288)
fontfamily or family = ['sans-serif']
fontname or name = DejaVu Sans
fontproperties or font or font_properties = sans\-serif:style=normal:variant=normal:weight=nor...
fontsize or size = 14.399999999999999
fontstyle or style = normal
fontvariant or variant = normal
fontweight or weight = normal
gid = None
horizontalalignment or ha = center
in_layout = True
label =
path_effects = []
picker = None
position = (0.5, 1.0)
rasterized = None
rotation = 0.0
rotation_mode = None
sketch_params = None
snap = None
stretch = normal
text = Customer Clusters (4)
transform = CompositeGenericTransform(     BboxTransformTo(    ...
transformed_clip_path_and_affine = (None, None)
unitless_position = (0.5, 1.0)
url = None
usetex = False
verticalalignment or va = baseline
visible = True
wrap = False
zorder = 3
```




```
# Create initial cluster indexes for Tenure and Bandwidth_GB_Year
X = churn_df.iloc[:, [35, 37]].values
```

```
# Calculate optimal amount of clusters with the elbow method
# Instantiate a (WCSS) Within Cluster Sum of Squares list
wcss = []

# Utilize a for loop to append values to WCSS list by iterating through kmeans datapoints
for i in range(2, 10):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Scree plot showing the optimal number of clusters
plt.plot(range(2, 10), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS List')
plt.savefig('tenure_v_bandwidth_plot1.jpg')
plt.show()
```




```
# Train the K-means model
kmeans = KMeans(n_clusters=2, init='k-means++', random_state=42)

# Dependent variable utilized to split customers into different clusters
y_kmeans = kmeans.fit_predict(X)
```

```
# Create Scatter plot using 4 clusters for Tenure v. Bandwidth_GB_Year
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 = 'green', label = 'cluster 2')

# Plot centroids for each cluster
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'yellow', label = 'Centroids')

# Plot Labels
title_obj = plt.title('Customer Clusters (2)')
plt.getp(title_obj)
plt.getp(title_obj, 'text')
plt.setp(title_obj, color='gray')

plt.xlabel('Tenure')
plt.ylabel('Bandwidth_GB_Year')

legend = plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.setp(legend.get_texts(), color='gray')

# Save plot
plt.savefig('tenure_vs_bandwidth_plot2.jpg')

# Display Plot
plt.show();
```

```
agg_filter = None
alpha = None
animated = False
bbox_patch = None
children = []
clip_box = None
clip_on = True
clip_path = None
color or c = black
contains = None
figure = Figure(432x288)
fontfamily or family = ['sans-serif']
fontname or name = DejaVu Sans
fontproperties or font or font_properties = sans\-serif:style=normal:variant=normal:weight=nor...
fontsize or size = 14.399999999999999
fontstyle or style = normal
fontvariant or variant = normal
fontweight or weight = normal
gid = None
horizontalalignment or ha = center
in_layout = True
label = 
path_effects = []
picker = None
position = (0.5, 1.0)
rasterized = None
rotation = 0.0
rotation_mode = None
sketch_params = None
snap = None
stretch = normal
text = Customer Clusters (2)
transform = CompositeGenericTransform(     BboxTransformTo(   ...
transformed_clip_path_and_affine = (None, None)
unitless_position = (0.5, 1.0)
url = None
usetex = False
verticalalignment or va = baseline
visible = True
wrap = False
zorder = 3
```

Part V: Data Summary and Implications

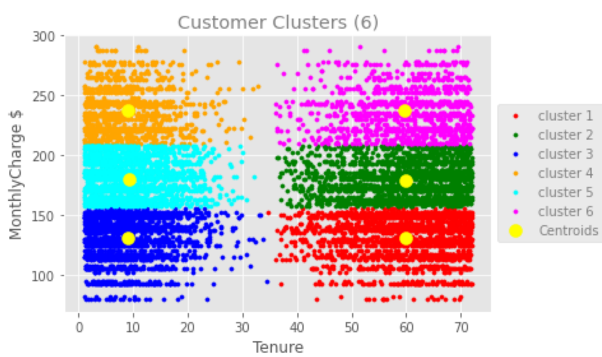
E1. Accuracy of Clustering Technique

As the K-means algorithm is an unsupervised algorithm, it would be difficult to quantify accuracy as a result. K-means is not strictly a classification tool, it is a clustering technique. The k-means algorithm it utilized to find groupings of datapoints and maximize cluster-distances.

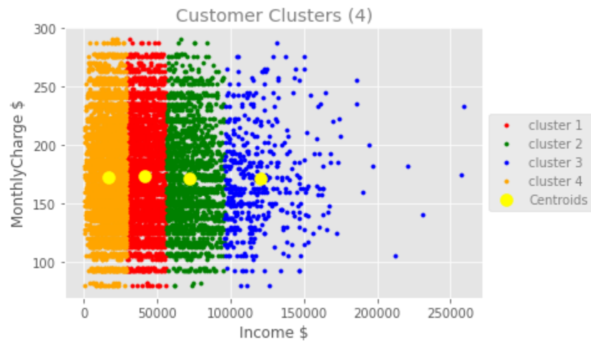
However, we can determine some degree of accuracy from the plotted graphs of calculations using the elbow method. By incorporating the elbow method into our calculations, we are able to determine the optimal number of clusters into which data can be clustered (GeeksforGeeks 2021). The elbow method is a popular method used to calculate the optimal value for the K variable.

To ensure accuracy to the highest degree, and based on calculations from the elbow method, we find that we should use 6 clusters for analyzing the features tenure and monthly charge, 4 clusters for analyzing income and monthly charge, and 2 clusters for analyzing tenure and bandwidth gigabytes per year.

E2. Results and Implications



Our first visualization comes from the k-means observation between tenure in months and monthly charge. The observation is in 2-dimensions and provides meaningful insight into the comparison of these two features. The elbow method recommends 6 clusters for optimal analysis, and from these neatly grouped clusters we can see 5 defined axes. We see two vertical axes correlating to Tenure in months, the first vertical axis on the 10-month mark, and the second vertical axis on the 60-month mark. This shows that we should be focusing on customer retention efforts during the 10th and 60th month of service. We also see three horizonal axes correlating to monthly charge, the first near \$130 price range, the second near \$175 range, and the third near \$240 range. These show the most popular average monthly charges across both the 10-month and 60-month tenure highlights.



Our second k-means observation comes from the comparison of the income and monthly charge features in the churn dataset. From the elbow method we have chosen to include 4 clusters for optimal analysis. From the visualization we can see one well defined axis and 4 distinct clusters. The axis common to all clusters is horizonal in the monthly charge category at the $175 price range. We have three well defined clusters, and a fourth more sporadic cluster with various outliers. The first three clusters are defined by the income range and start from 0 to $40,000, the second cluster from $40,000 to $60,000, the third cluster from $60,000 to $90,000, and the fourth cluster from $90,000 to $150,000 which also includes outliers up to $250,000.

Our last observation comes from the comparison of the tenure by month and bandwidth gigabytes per year features. The elbow method recommends two clusters for optimal analysis. In our visualization we can see one well defined axis. There appears a linear correlation between tenure and bandwidth gigabyte per year consumed. Once again, as in our first analysis there appears two centroids at the 10-month and 60-month tenure marks. We see a positive linear correlation as the amount of data consumed increases, so does tenure. The two centroids for each cluster appear at the 1500 and 5500 data gigabyte consumption marks.

E3. Limitation

The greatest limitation of the data analysis is the limited size and scope of the provided WGU Churn dataset. In a real-world scenario, a telecommunications company would have millions or at the least hundreds of thousands of customers. While we can gain meaningful insights from available data, not having a dataset with a realistic size of customers limits the analysis and data techniques available such as regression, classification, and machine learning models. However, the advanced data acquisition assignment provides meaningful and excellent practice in database management and data manipulation using industry approved data analytics software. Another limitation of the experience is the inability to communicate with the stakeholders at the Telcom company or any subset such as data engineers, executives, or even customer representatives. Without communication with Telcom employees it is hard to gain clarification about data or to provide targeted insights into churn metrics. Access to personnel and meetings with decision makers would provide a better alignment for reaching data analytics goals and providing the most benefit from actionable insights.

E4. Course of Action

From our k-means Clustering Data Mining analysis we gain several useful insights for stakeholders and decision makers. One insight from our first cluster analysis is the company retention department should focus on customers reaching the 10-month and 60-month tenure marks as this comprises the average majorities of customers. We also find the marketing team should focus on creating incentives for customers at the $130, $175, and $240 monthly rate plans.

In our second cluster analysis we find that the $175 price plan is the most popular among all income ranges. This insight should prove useful for the marketing team to ensure that this monthly price plan includes incentives that are attractive to low, middle, and high-income earners.

From our third cluster analysis we confirm an observation seen in exploratory analysis. There is a linear correlation between tenure and bandwidth consumed each year. The more data customers use annually, the more likely they are to remain customers.

From our exploratory analysis we strongly recommend ensuring that customer issues are resolved quickly and that the equipment provided is reliable and of quality, with fewer equipment replacements. Additionally, the data shows that customers with more services added to their accounts such as tech support and online backups are more likely to stay with the company so these optional services should be advertised and promoted in future marketing campaigns.

Part VI: Demonstration

F. Panopto Recording

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=34ae4737-aa0f-4fd5-aa80-ae5b00a61371

G. Sources for Third-Party Code

Cluster Analysis in python course. DataCamp. (n.d.). Retrieved March 17, 2022, from https://www.datacamp.com/courses/cluster-analysis-in-python

K-means clustering with scikit-learn. DataCamp Community. (n.d.). Retrieved March 17, 2022, from https://www.datacamp.com/community/tutorials/k-means-clustering-python

Elbow method for optimal value of K in kmeans. GeeksforGeeks. (2021, February 9). Retrieved March 17, 2022, from https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

I. Sources

Sklearn.cluster.kmeans. scikit. (n.d.). Retrieved March 17, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Garbade, D. M. J. (2018, September 12). *Understanding K-means clustering in machine learning*. Medium. Retrieved March 14, 2022, from https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1