WGU D212 PA 1

Andrew Shrestha

Part I: Research Question

A1. Question: Are we able to accurately predict which customers are at risk of Churn? Can we identify the significant variables that predict the churn outcome? This will be done through the use of K Means Clustering Technique.

A2. Objectives and Goals: There are 2 very important reasons why a company would benefit from tracking its churn rate. First, the loss of customers is equivalent to loss of revenue, and due to this being a driving factor in a company's future growth and development, becomes dire to prevent. Secondly, it is generally more expensive to gain new customers than it is to maintain existing ones, therefore it is in the company's and shareholder's best interest to allocate resources to prevent customer churn **(Zendesk 2021).**

With the knowledge of the significant variables identified and our prediction of which customers will churn, companies will have a strong sense of where to allocate their resources to ensure that customer churn is reduced.

Part II: Method Justification

B1. Explanation of Classification Method using K-Means Clustering

The idea of the K Means Clustering algorithm is to partition our data set into distinct subgroups where each data point belongs to one specific cluster. This will in turn keep data points within clusters similar, while ensuring maximum differences between clusters. This would be done by running iterations until the sum of squared distance between data points and cluster centroids are at a minimum. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

What this means for our project is that we cluster our telecom customers so that they are in groups where characteristics and variable features are similar. Using common knowledge, it is possible to have some predicted outcomes already expected for clustering such as StreamingMovies, StreamingTv, and Bandwidth Usage per Year customer behaviours as an example.

Understanding customers that are places in the same group will give us an idea of which characteristics are more prevalent in these customers who either decide to churn or not. This is results in a very easy and insightful algorithm.

B2. Summary of One Assumption for K-Means Clustering

The assumption of importance is that the variance of the distribution of each attribute is spherical. The reason for this assumption is that the algorithm clusters based on centroids and thus only works when clusters are grouped spherically. Even if there are clusters that are visibly noticeable through basic visual observations, the algorithm will not run correctly if this assumption is violated. http://varianceexplained.org/r/kmeans-free-lunch/#:~:text=k%2Dmeans%20assume%20the%20variance,then%20k%2Dmeans%20will%20fail.

B3. Package and Library List

In this Project, I will be leveraging the use of Python Programming language as it is a general purpose and production ready language with clear and easily interpretable syntax. On a personal note, I have used Python before in previous Data Science Certifications and have introductory knowledge on the language.

Libraries and Packages that will support the data cleaning process will be as follows:

Pandas: create and store the data in a tabular form which makes it extremely effective for data manipulation. Pandas is also able to detect missing values and replace them with an "NAN" or "NA" and is useful in manipulating both numerical and string values.

Numpy: Grants us the use of multi dimensional arrays and computational functions for mathematical and statistical analysis

Matplotlib: Grants us the use of data visualization and creates various types of plots and graphs with the goal of representing data. Also very handy for visually providing insights into identifying potential outliers or missing data.

Seaborn: Based on Matplotlib, it gives us advanced data visualization with more informative statistical graphics

Scikit – Learn: predictive analysis with useful features such as Classifications, Regression, and Clustering.

SciPy: mathematical analysis for integration, optimization, algebra, and statistics

Part III: Data Preparation

C1. One Preprocessing Goal Relevant to Classification

The one preprocessing goal for this analysis is to ensure that the binary categorical variables of Yes/No or Male/Female are all converted to binary dummy variables with results of 1 or 0. This will be the most important step to prepare our data for the analysis.

C2. Data Set Variables and Classification

Initial Data Set Continuous Variables:

1) Children
2) Income
3) Outage_sec_perweek

4) Email
5) Contacts
6) Yearly_equip_failure
7) Tenure
8) MonthlyCharge
9) Bandwidth_GB_Year

Initial Data Set Binary Categorical Variables:

1) Techie (Yes/No)
2) Contract (Month-to-Month/One Year/Two Year)
3) Port_modem(Yes/No)
4) Tablet (Yes/No)
5) InternetService (DSL/Fiber Optic/None)
6) Phone (Yes/No)
7) Multiple (Yes/No)
8) OnlineSecurity (Yes/No)
9) OnlineBackup (Yes/No)
10) DeviceProtection (Yes/No)
11) TechSupport (Yes/No)
12) StreamingTV (Yes/No)
13) StreamingMovies (Yes/No)

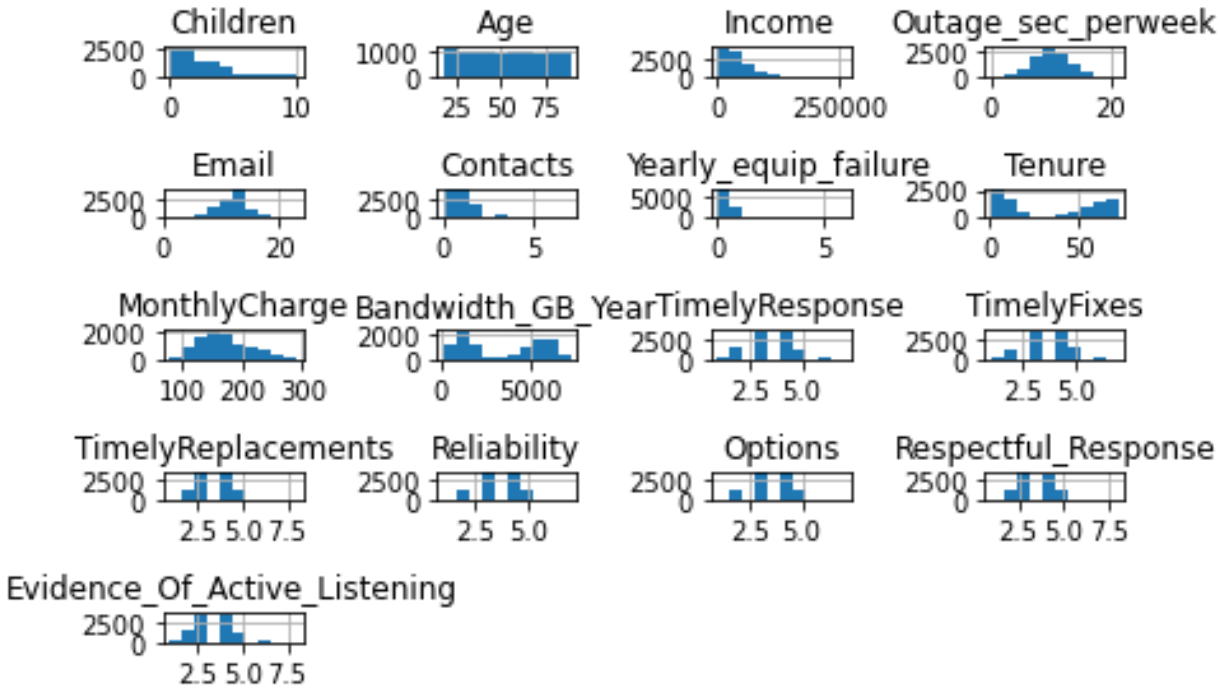
Initial Data Set Discrete Numerical Variables:

1) Item1: Timely Response
2) Item2: Timely Fixes
3) Item3: Timely Replacements
4) Item4: Reliability
5) Item5: Options
6) Item6: Respectful Response
7) Item7: Courteous Exchange
8) Item8: Evidence of Active Listening

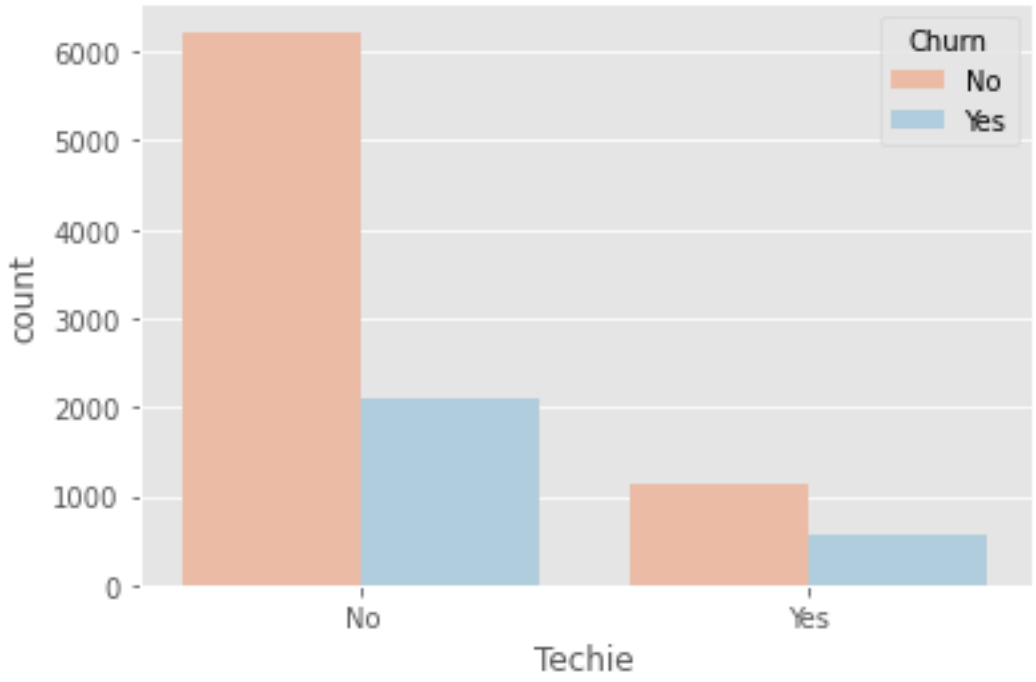
C3. Data preparation code

```
#Importing the standard Python libraries for Analysis and Visualizations
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


#Load the CSV data set file into our Pandas Dataframe
churn_clean_df =
pd.read_csv(r"C:\Users\andre\OneDrive\Desktop\churn_clean.csv")


#now we want to get a surface level understanding of our data that we just
imported via basic statistical analysis
churn_clean_df.shape
churn_clean_df.columns

churn_clean_df.head()
```

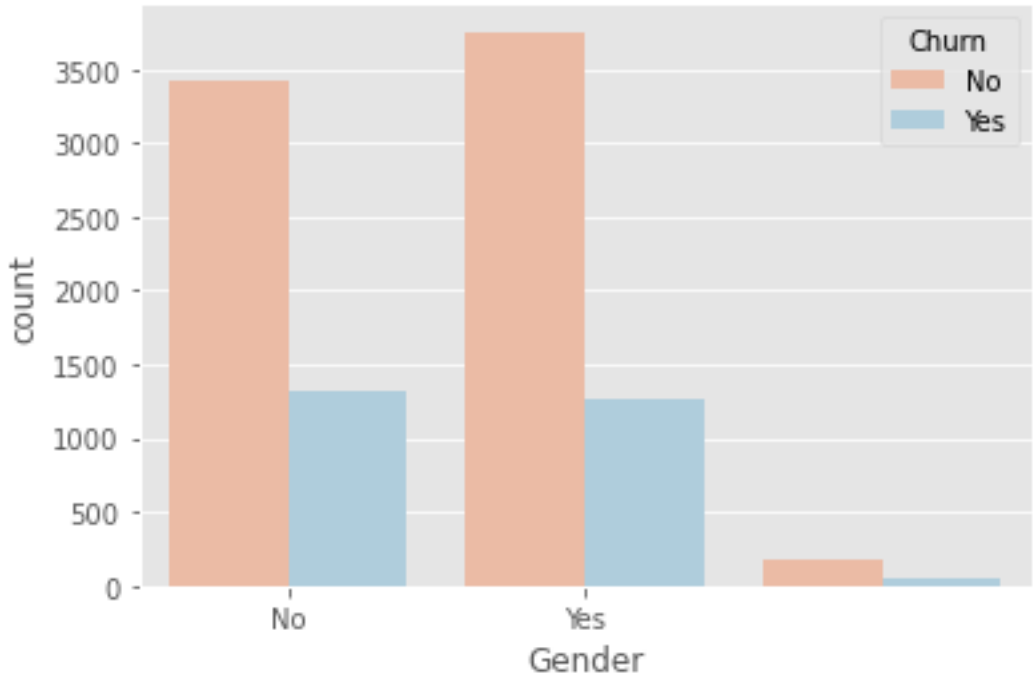
```
churn_clean_df.info

churn_clean_df.describe()

churn_clean_df.dtypes



#we want to now rename the survey questions from basic numerical labels to
more detailed labels in order to avoid confusion and make it clearer in the
analysis process
churn_clean_df.rename(columns = {'Item1':'TimelyResponse',
                     'Item2':'TimelyFixes',
                      'Item3':'TimelyReplacements',
                      'Item4':'Reliability',
                      'Item5':'Options',
                      'Item6':'Respectful_Response',
                      'Item7':'Courteous_Exchange',
                      'Item8':'Evidence_Of_Active_Listening'},
          inplace=True)


#let us now create some visualizations of the significant continuous and
categorical variables present in the data set via hist plots
churn_clean_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
          'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
          'Bandwidth_GB_Year',
'TimelyResponse','TimelyFixes','TimelyReplacements','Reliability', 'Options',
'Respectful_Response', 'Evidence_Of_Active_Listening']].hist()
plt.tight_layout()
```



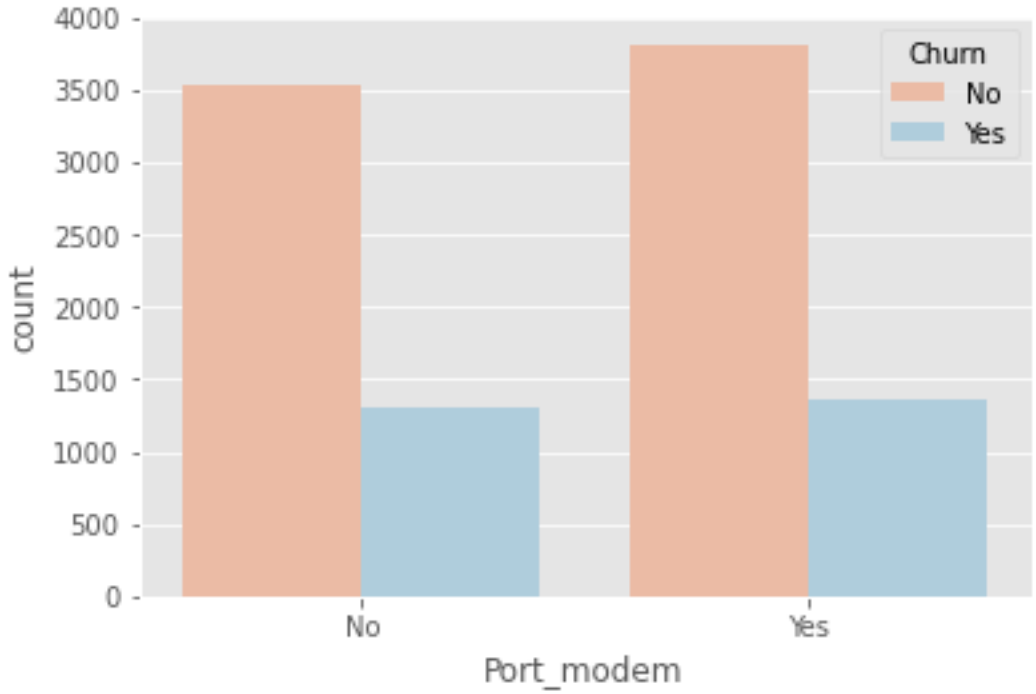

```
#Now using ggplot, we can also see the bivariate visualizations between our
categorical binary variables and that of 'Churn'
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Techie', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



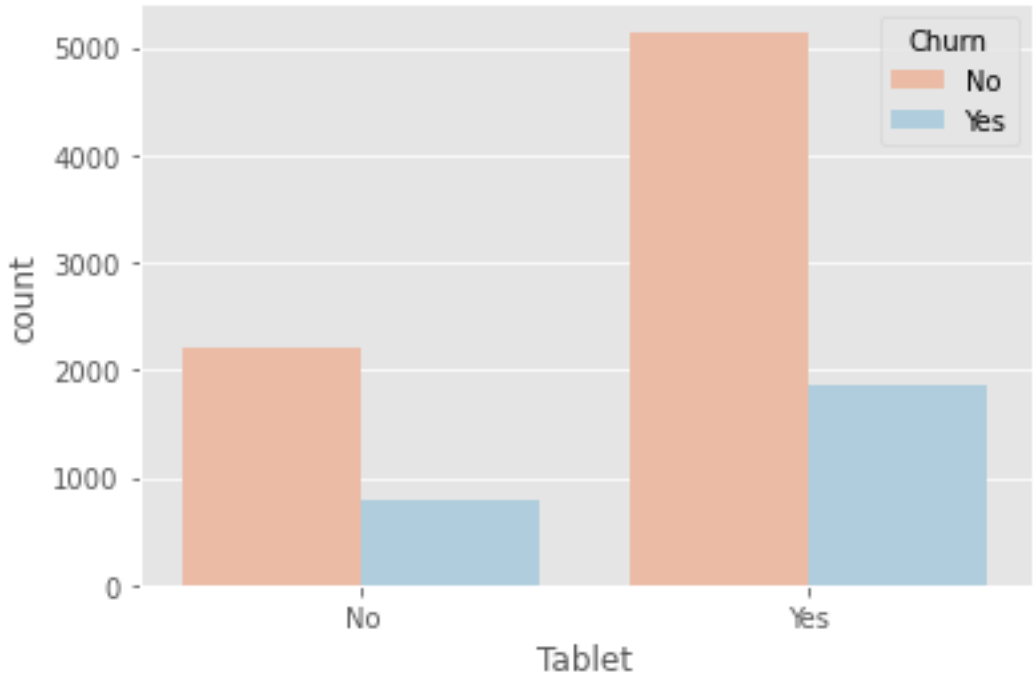

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Gender', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



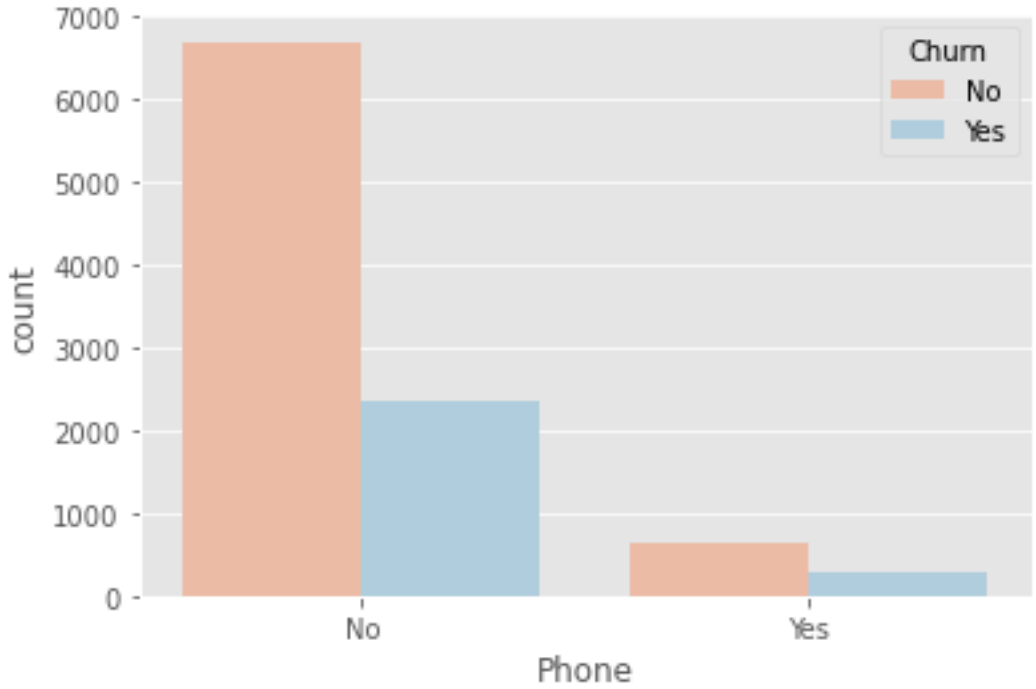
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Port_modem', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



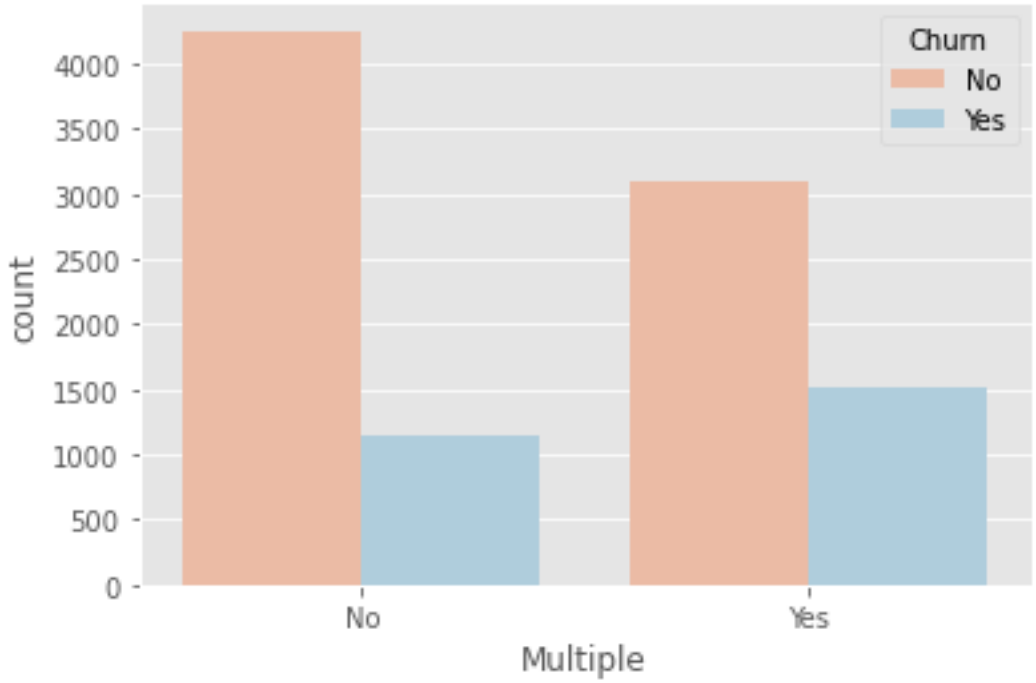

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Tablet', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



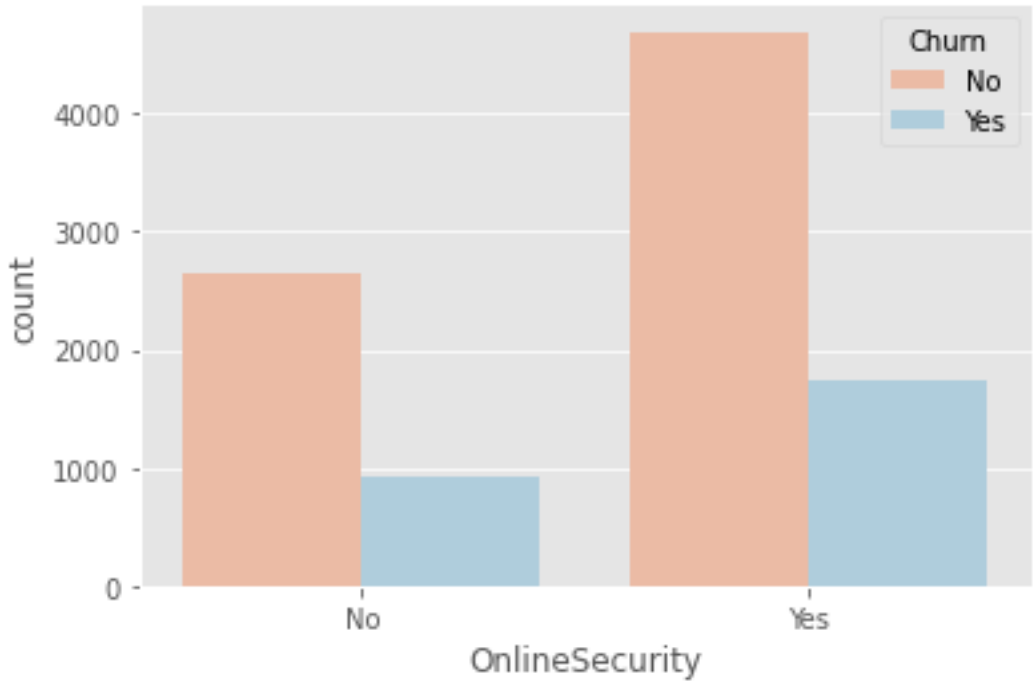

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Phone', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



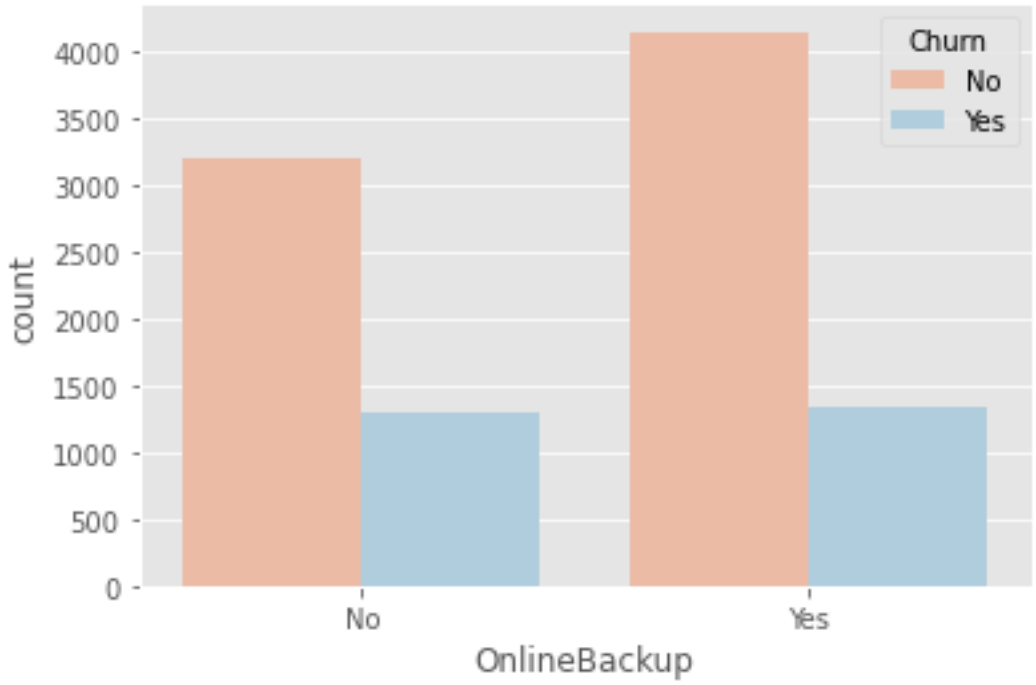
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Multiple', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



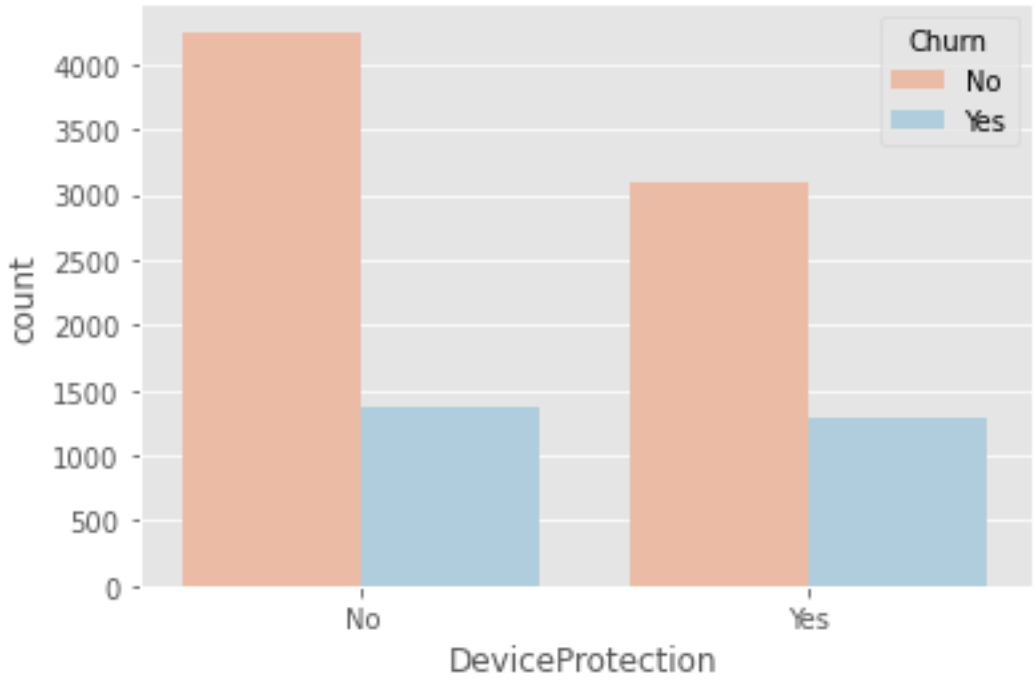
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='OnlineSecurity', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



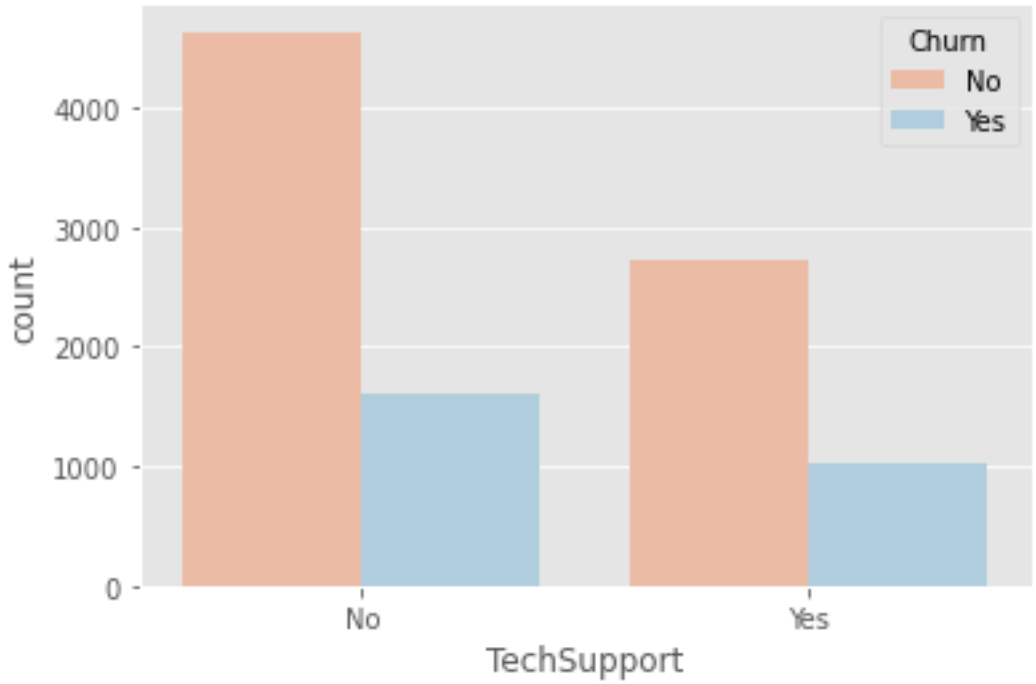
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='OnlineBackup', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



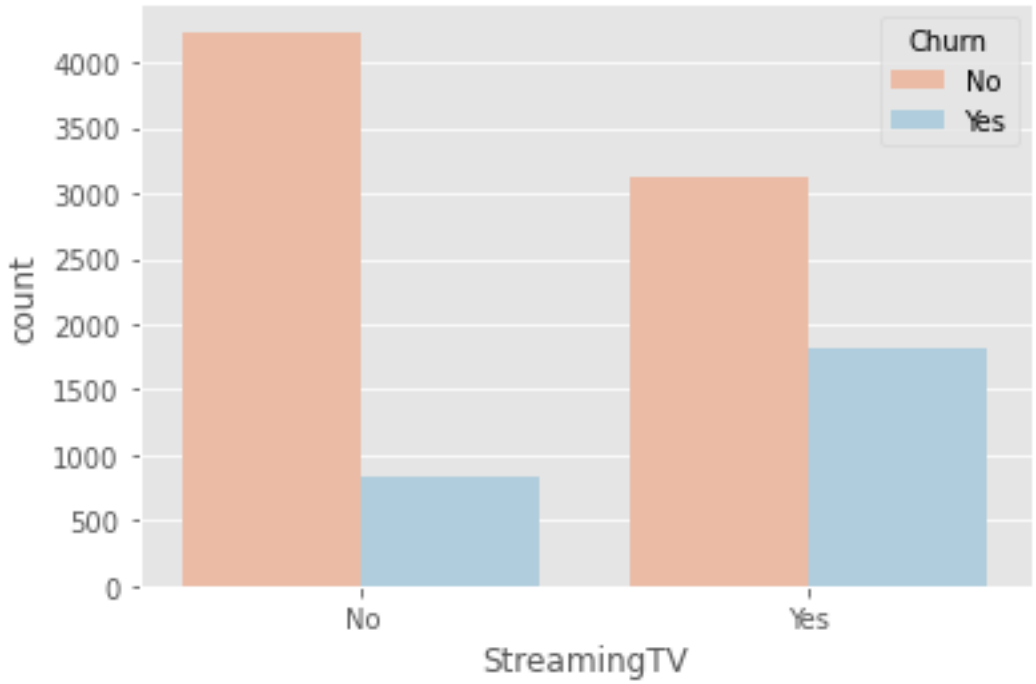
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='DeviceProtection', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



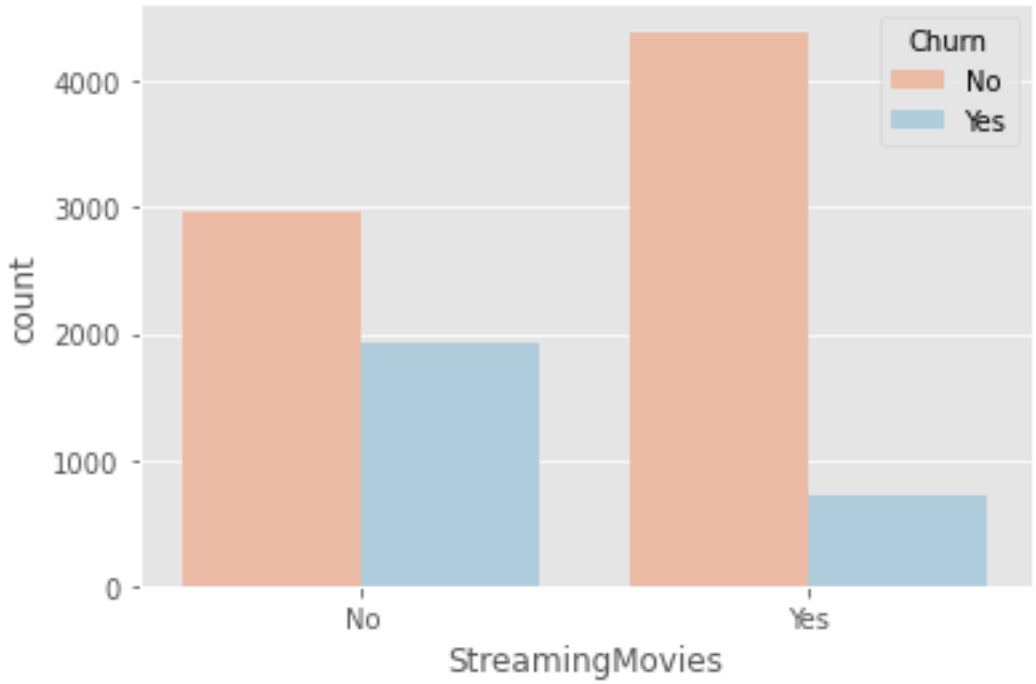
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='TechSupport', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



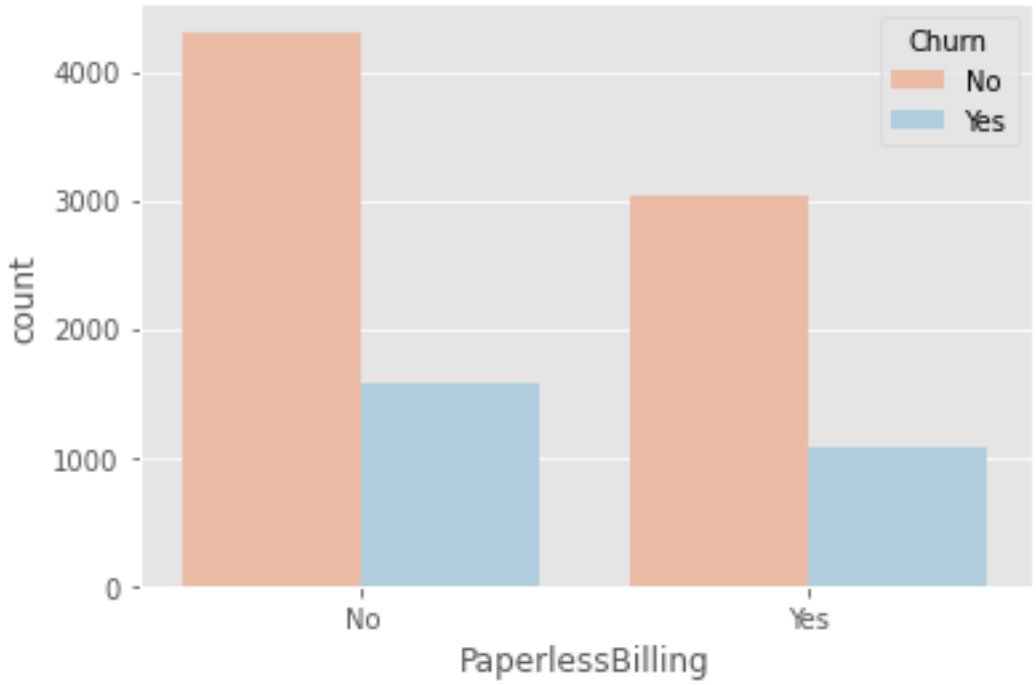
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='StreamingTV', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='StreamingMovies', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='PaperlessBilling', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```




```
#Now we will see if there are any anomolies in the form of missing data in the
given data set before applying it to the analysis process
```

```
data_nulls = churn_clean_df.isnull().sum()
print(data_nulls)
```

```
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
TimelyResponse                 0
TimelyFixes                    0
TimelyReplacements             0
Reliability                    0
Options                        0
Respectful_Response            0
Courteous_Exchange             0
Evidence_Of_Active_Listening   0
```

```
#Now we have to encode our binary categorical variables with a numerical
value of either 1 or 0. This is so that we are able to properly process this
in our analysis step
churn_clean_df ['DummyGender'] = [1 if v =='Male' else 0 for v in
churn_clean_df['Gender']]
churn_clean_df ['DummyChurn'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Churn']]
churn_clean_df ['DummyTechie'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Techie']]
churn_clean_df ['DummyContract'] = [1 if v =='Two Year' else 0 for v in
churn_clean_df['Contract']]
churn_clean_df ['DummyPort_modem'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Port_modem']]
churn_clean_df ['DummyTablet'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Tablet']]
churn_clean_df ['DummyInternetService'] = [1 if v =='Fiber Optic' else 0 for
v in churn_clean_df['InternetService']]
churn_clean_df ['DummyPhone'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Phone']]
churn_clean_df ['DummyMultiple'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Multiple']]
churn_clean_df ['DummyOnlineSecurity'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['OnlineSecurity']]
churn_clean_df ['DummyOnlineBackup'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['OnlineBackup']]
churn_clean_df ['DummyDeviceProtection'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['DeviceProtection']]
churn_clean_df ['DummyTechSupport'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['TechSupport']]
churn_clean_df ['DummyStreamingTV'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['StreamingTV']]
churn_clean_df ['DummyStreamingMovies'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['StreamingMovies']]
churn_clean_df ['DummyPaperlessBilling'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['PaperlessBilling']]


#Now we will drop the original (Yes/No) categorical variables as we
essentially already created a duplicate of them with binary 0 or 1 values
churn_clean_df = churn_clean_df.drop (columns = ['Gender',
'Churn','Techie','Contract','Port_modem','Tablet','InternetService','Phone','
Multiple','OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','S
treamingTV','StreamingMovies','PaperlessBilling'])


churn_clean_df.head()


#Remove less significant columns in order to reduce dataset and make it
easier for analysis
```

```
#Remove less significant columns in order to reduce dataset and make it easier
for analysis

churn_clean_df = churn_clean_df.drop (columns = ['CaseOrder',
'Customer_id','Interaction','UID','City','State','County','Zip','Lat','Lng','Po
pulation','Area','TimeZone','Marital','PaymentMethod','Job'])


churn_clean_df.head()


#Finally we Extract the cleaned data set
churn_clean_df.to_csv('data_churn_clean_prepared.csv')
```

C3. Copy of Cleaned Data Set

Copy of the Cleaned Data Set saved as 'data_churn_clean_prepared.csv' is attached

Part IV: Analysis

D1. Analysis and Intermediate Calculations

Utilizing the KMeans method, we take three pairs of features/variables that are related to the Churn result and create our analysis based on the clustering techniques below. With this we are able to gain a greater insight into the customer base according to these selected features.

1) Tenure and MonthlyCharge

2) Income and MonthlyCharge

3) Tenure and Bandwidth_GB_Year

Scree elbow plot method were used along with the clustering method in order to identify the optimum number of clusters for the above three pairs of data.

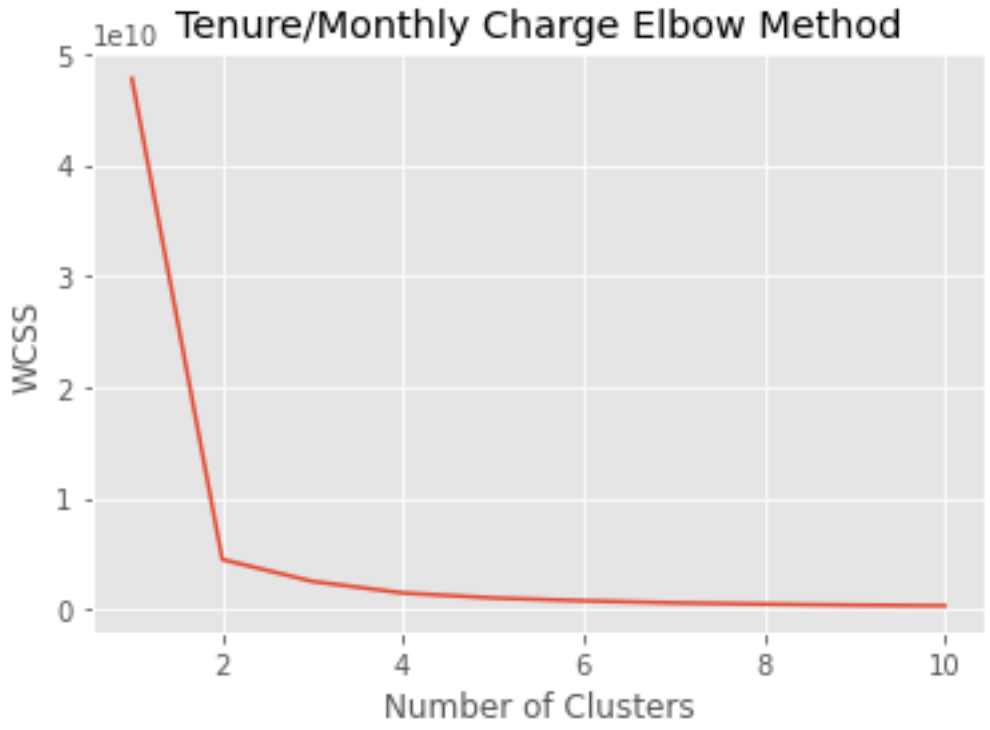
D2. Code for Clustering Analysis

```
# Reload the prepared data set for our analysis and import the necessary Kmeans
libraries
churn_df = pd.read_csv('data_churn_clean_prepared.csv')

from sklearn.cluster import KMeans

plt.style.use('ggplot')


# List features for analysis
features = (list(churn_df.columns[:-1]))
print('Features for analysis include: \n', features)
```

```
Features for analysis include:
 ['Unnamed: 0', 'Children', 'Age', 'Income', 'Outage_sec_perweek',
'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
'Bandwidth_GB_Year', 'TimelyResponse', 'TimelyFixes',
'TimelyReplacements', 'Reliability', 'Options', 'Respectful_Response',
'Courteous_Exchange', 'Evidence_Of_Active_Listening', 'DummyGender',
'DummyChurn', 'DummyTechie', 'DummyContract', 'DummyPort_modem',
'DummyTablet', 'DummyInternetService', 'DummyPhone', 'DummyMultiple',
'DummyOnlineSecurity', 'DummyOnlineBackup', 'DummyDeviceProtection',
'DummyTechSupport', 'DummyStreamingTV', 'DummyStreamingMovies']
```

Features for Tenure and MonthlyCharge

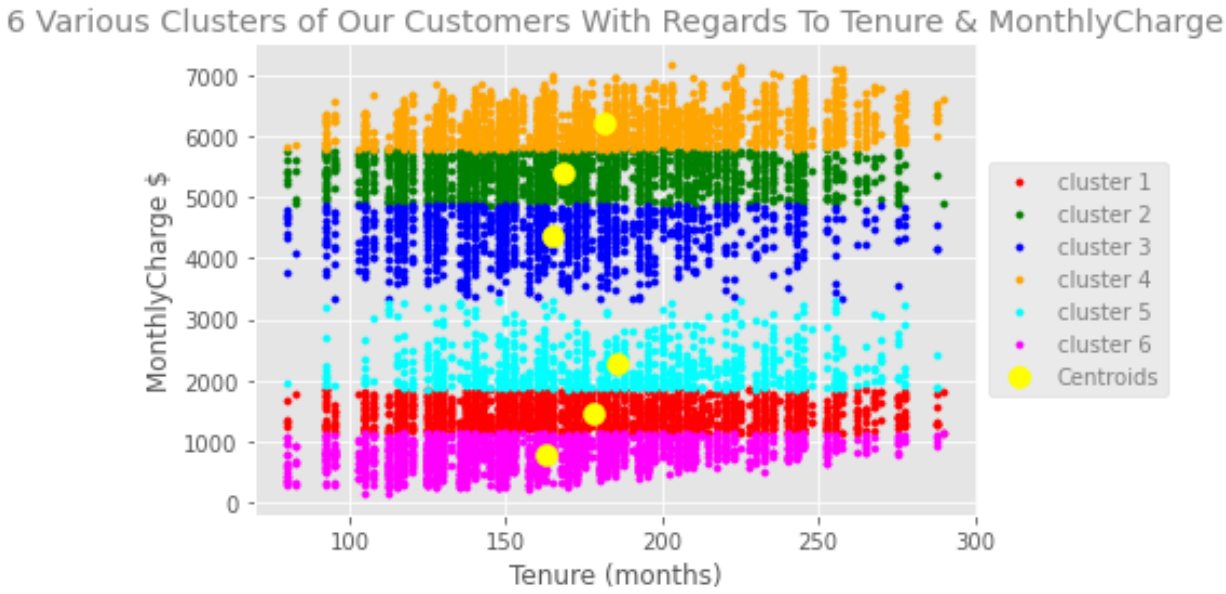
```
# Select the features of Tenure and MonthlyCharge for our initial portion of
clustering
X = churn_df.iloc[:, [9, 10]].values


# Use the elbow methodology to obtain the optimal number of clusters
# Also Creating a Within Cluster Sum of Squares (WCSS) list to measure
variability within each cluster.
wcss = []

# Write a for loop to write values to wcss list by iterating through kmeans
objects
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Scree plot the optimal number of clusters
plt.plot(range(1, 11), wcss)
plt.title('Tenure/Monthly Charge Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```




```
# Train the K-means model on the dataset created using the optimal number 6 as
shown with the elbow bend above. We now train the
#model on the dataset with the number of clusters = 6
kmeans = KMeans(n_clusters=6, init='k-means++', random_state=42)

# Build the dependent variable to split customers into the differing clusters
y_kmeans = kmeans.fit_predict(X)


# Our y_kmeans gives us the different clusters corresponding to X. We can now
plot all of the clusters using matplotlib.
print(y_kmeans)
# Scatter plot the 6 clusters for our Tenure/ 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 of each cluster
```

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

# Generate plot description
title_obj = plt.title('6 Various Clusters of Our Customers With Regards To
Tenure & MonthlyCharge')
plt.getp(title_obj)
plt.getp(title_obj, 'text')
plt.setp(title_obj, color='gray')

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

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


# Plot it
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
```

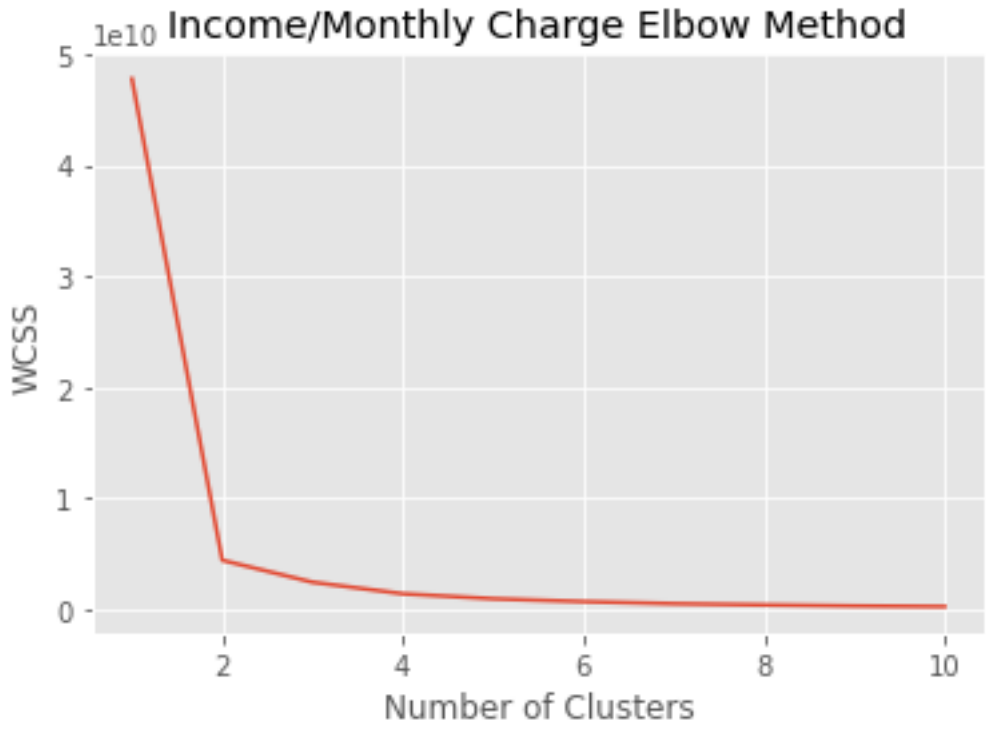
```
text = 6 Various Clusters of Our Customers With Regards T...
transform = CompositeGenericTransform(     BboxTransformTo(    ...
transformed_clip_path_and_affine = (None, None)
unitless_position = (0.5, 1.0)
url = None
usetex = False
verticalalignment or va = baseline
visible = True
wrap = False
zorder = 3
```



Features for Income and MonthlyCharge

```
# Select the features of Income and MonthlyCharge for our initial portion of clustering
X = churn_df.iloc[:, [4, 10]].values


# Use the elbow methodology to obtain the optimal number of clusters
# Also Creating a Within Cluster Sum of Squares (WCSS) list to measure variability within each cluster.
wcss = []

# Write a for loop to write values to wcss list by iterating through kmeans objects
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Scree plot the optimal number of clusters
plt.plot(range(1, 11), wcss)
plt.title('Income/Monthly Charge Elbow Method')
```

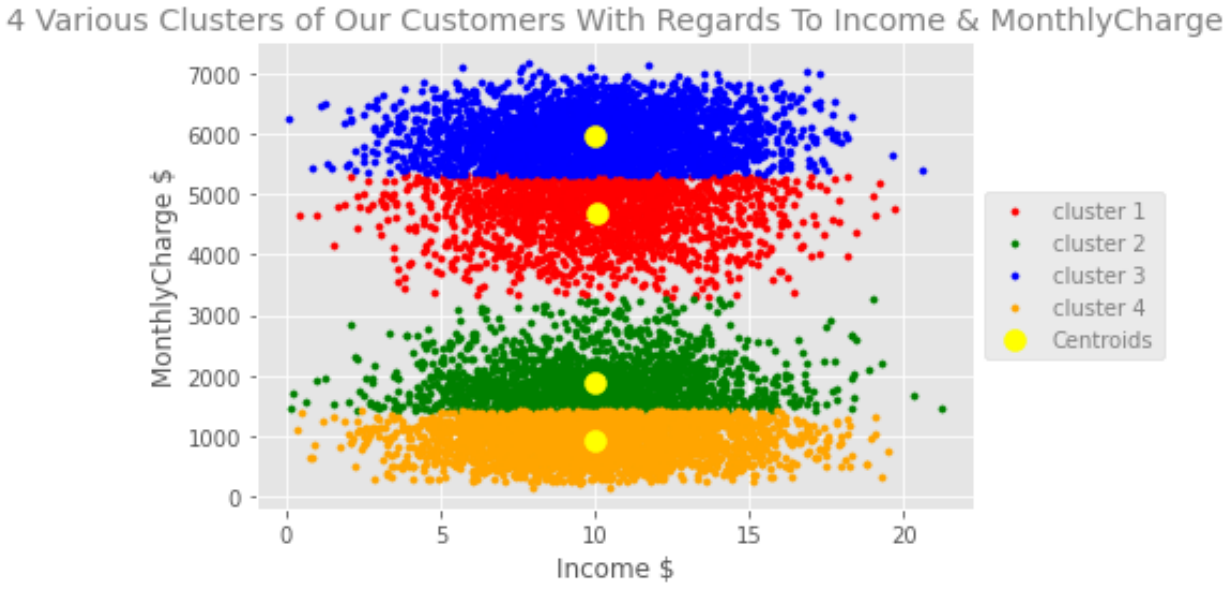
```
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```




```
# Train the K-means model on the dataset created using the optimal number 4 as
shown with the elbow bend above. We now train the
#model on the dataset with the number of clusters = 4
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)

# Build the dependent variable to split customers into the differing clusters
y_kmeans = kmeans.fit_predict(X)


# Scatter plot the 4 clusters for our Income/ 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 of each cluster
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s =
100, c = 'yellow', label = 'Centroids')
```

```
# Generate plot description
title_obj = plt.title('4 Various Clusters of Our Customers With Regards To
Income & MonthlyCharge')
plt.getp(title_obj)
plt.getp(title_obj, 'text')
plt.setp(title_obj, color='gray')

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

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


# Plot it
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 = 4 Various Clusters of Our Customers With Regards T...
    transform = CompositeGenericTransform(     BboxTransformTo(   ...
    transformed_clip_path_and_affine = (None, None)
    unitless_position = (0.5, 1.0)
    url = None
```

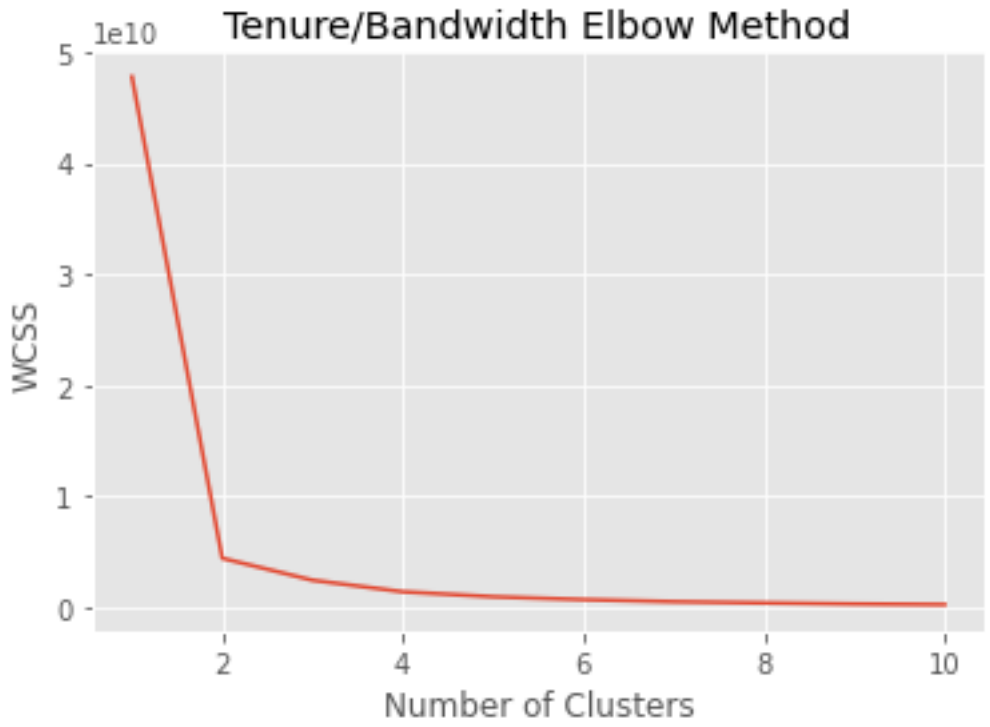
```
    usetex = False
    verticalalignment or va = baseline
    visible = True
    wrap = False
    zorder = 3
```



Features for Tenure and Bandwidth

```
# Select the features of Tenure and Bandwidth for our initial portion of
clustering
X = churn_df.iloc[:, [10, 12]].values


# Use the elbow methodology to obtain the optimal number of clusters
# Also Creating a Within Cluster Sum of Squares (WCSS) list to measure
variability within each cluster.
wcss = []

# Write a for loop to write values to wcss list by iterating through kmeans
objects
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Scree plot the optimal number of clusters
plt.plot(range(1, 11), wcss)
plt.title('Tenure/Bandwidth Elbow Method')
plt.xlabel('Number of Clusters')
```

```
plt.ylabel('WCSS')
plt.show()
```




```
# Train the K-means model on the dataset created using the optimal number 2 as
shown with the elbow bend above. We now train the
#model on the dataset with the number of clusters = 2
kmeans = KMeans(n_clusters=2, init='k-means++', random_state=42)

# Build the dependent variable to split customers into the differing clusters
y_kmeans = kmeans.fit_predict(X)


# Scatter plot the 2 clusters for our Tenure/ Bnadwidth

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

# Generate plot description
title_obj = plt.title('2 Various Clusters of Our Customers With Regards To
Tenure & Bandwidth')
plt.getp(title_obj)
plt.getp(title_obj, 'text')
```

```
plt.setp(title_obj, color='gray')

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

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


# Plot it
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 = 2 Various Clusters of Our Customers With Regards T...
    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

Clustering is an inherently complex task and thus, it is very important to be able to assess and evaluate the clustering results. https://towardsdatascience.com/are-the-clusters-good-1567de6a9524

Two clustering evaluations that we can look at are the Clustering Tendency and Clustering Quality (Cohesion and Separation).

1. Clustering Tendency:

This is a measure of whether or not the data set and cluster analysis is able to give meaningful clusters (non-random data). In our scenario, there are a lot of data points that are binary in nature and uniformly distributed data. An example of this would be the discrete numeric values of our survey results or the "Yes/No" outputs for our Dummy Variables. Thus, we choose non-uniform features in or data set such as: Income, Tenure. MonthlyCharge, and Bandwidth_GB_Year. Due to their non-uniform nature, they are able to give us a true representation of meaningful clusters generated through our analysis.

2. Clustering Quality:

This is a measure of a combination of Cohesion and Separation. Cohesion deals with observations within a cluster to be as close as possible, while Separation deals with distance between two clusters. In our scenario, we see that our data points within clusters and not tightly centered around their centroids. This is because there are a lot of factors that go into a detailed profile for our customers, thus, the data for each cluster behaves more sectionally rather than discrete cluster points.

E2. Results and Implications

The results are broken down into the three pairs of significant features that were relevant to customer churn. Through our kMeans clustering analysis, we found the following:

1) Tenure & MonthlyCharge:

This was shown to have 6 optimal clusters. When looking at the data clusters, it seems that the customers who display the characteristic of lower MonthlyCharge tend lead to an increased tenure, thus less likely to churn. This could make sense if we take the majority of customers as being conservative in their disposable income spending, due to the fact that more services or features on top of the basic services can be seen as luxury and not necessary for the majority. As extra services or costs are added, they may leave to seek out more affordable prices.

Therefore, when it gets to higher MonthlyCharges, one way is to implement discounts to effectively lower the cost for customers and offer them better deals, thus, enticing them to stay longer and increase the tenure with the telecom company.

2) Income & MonthlyCharge:

This was shown to have 4 optimal clusters. When looking at the data clusters, it was interesting to see that regardless of the income level, the spending of customer income with MonthlyCharge was quite very uniform across the income ranges. This is contradictory to an surface level hypothesis that higher income would equal higher MonthlyCharges and vice versa.

Implications of this show that we have no problem getting lower income customers to spend more with us, however, we should be focusing on increasing the higher income customers to increase their MonthlyCharges more significantly. This could be done by using a tier rewards system to get more of their disposable income into the company and make them feel more special and rewarded with limited offered services.

3) Tenure & Bandwith_GB_Year:

This was shown to have only 2 optimal clusters. When looking at the data clusters, it seems that customers who end up utilizing low rates of Bandwidth tend to also have a lower tenure and vice versa. This could make sense as the more people are happy with the services and speed of the Bandwidth provided, the longer they will stay. Implications of this show that perhaps we can offer a reduced trial for more bandwidth for a certain period of time to allow the customers to build up rapport and need for the bandwidth usage, then after that they can choose to stay and pay more for the amount of Bandwidth they are use to having. This could increase both Tenure and Bandwidth_GB_Year in the long run.