PERFORMANCE ASSESSMENT_ D212

OFM3 TASK 1: CLUSTERING TECHNIQUES METHODS

March 7th,2022

Table of Contents

Performance Evaluation of OFM3: CLUSTERING TECHNIQUES, D212

Name: Rekha Alle
Student ID: 000778673
Course: Masters Data Analytics
Date: 03/07/2022
Program Mentor: Kesselly kamara
Contact: 385-428-4395
Email: Kesselly.kamara@wgu.edu

Part I. Introduction to Scenario

Understanding customers is one of the most important aspects of customer relationship management that directly influences a company's long-term success. When a corporation has a greater understanding of its consumers' traits, this could good target promotion and advertising campaigns for them, resulting in higher long-term earnings.

You operate as an investigator for a telecoms business that wants to understand more about its customers' characteristics. You've been given with conducting market basket research on customer research to discover critical relationships between consumer purchases, enabling for better operational and organizational selection.

A1. Analytical Question:

Which of your clients' primary characteristics indicate that they are at significant risk of churn? As a result, which clients are likely to leave? In other words, can we use deep classification of data mining to comprehend business customers and identify trends specific to churning customers?

The clustering technique methods of K-means will be used to solve this analysis.

A2. Goals and Objectives:

Everybody in the organization will profit from recognizing, with some degree of certainty, which customers will be able to churn, since it will give importance to selling enhanced services to consumers with these traits and previous user experiences. This data analysis' purpose is to give numerical information to company stakeholders to assist them better understand their customers.

Part II. Justification for the method

B1. Assumptions Summary: Clustering Methodology

We are not attempting to forecast a result y relying on a variable (x) X in k-means clustering. In general, we're looking for trends in our data. To be even more particular, we establish a variable to recognize those trends. We make it so that each of the potential relying on variable's values corresponds to one of the relying variable's classes (SuperDataScience). To be clear, there is not a prior primary outcome.

As an outcome, we're trying to "cluster" our subscribers are divided into groups based on common features such as annual bandwidth usage or duration with the organization. "[A] decent clustering solution is one that finds clusters where the observations inside each cluster are more similar than the clusters themselves," Jeffares says (Jeffares, p. 1)[1].

Hierarchical and k-means clustering approaches are the two clustering strategies presented in this task. According to a Data Camp course, "[a] key downside of runtime [is] hierarchical clustering", (Daityari, p. 1)[2]. While the dataset we're looking at isn't especially enormous, neither is the machine I'm using. One of the reasons we use the k-means algorithm is for this reason.

To choose k-means rather structured, we also analyzed the dataset size and patterns. Customer churn isn't about separating countries in soccer matches or creating a dendrogram, after all. What we need to show stakeholders is which consumer groups (clusters) are comparable. And, of course, how similar/different our consumer groups are, as well as how tightly/loosely packed they are (market segments).

The creative phase of investigation is this step. At this phase in the project, some trial and error are allowed.

Ultimately, we anticipate seeing consumer attrition having lower periods, using less of the supplied telecom services with the organization or perhaps we can learn from the survey results that consumer who churned were less satisfied and gave the company's customer service a lower rating, whilst customers who stayed loyal gave it a higher rating.

To analyze the data using Kmean technique we:

- First, we cleaned the dataset and extracted and loaded as “churn_prepared_kmeans.csv”
- Once loaded, we import Kmeans from Sklearn.cluster
- And then we use the Elbow method to find the optimum number of clusters and made a list of WCSS (Within Cluster Sum of Squares) by looping through kmean objects.
- And we plot the wcss list as 'churn_scree_tenure_v_monthly-charge.jpg'

- Once the dataset is ready, train the K-mean model on that dataset and using fit_predict method to divide customers into various clusters, we create the dependent variable.
  - Then we visualize the clusters using plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'yellow', label = 'Centroids') for:
    - **Tenure vs Monthly charges:** 6 Clusters of customers
    - **Income vs Monthly charges**: 4 Clusters of customers
    - **Tenure vs Bandwidth**: 2 Clusters of customers
- and plot centroids of each cluster
- Once we plot the centroids, we generate plot description like get the title property handler, set the color of the title, etc.…

B2. Appropriate Methodology:

As VanderPlas notes out, a fundamental assumption in K-means clustering is that the "cluster center," or centroid, is all points within the arithmetic means (VanderPlas, p. 463)[3] or "belonging to" a cluster.

"We wish to locate the centroid C that eliminates" the distortion, Jeffares demonstrates.

The Inside Cluster Sum of Squares (WCSS) is the variability measure of the data within each cluster, where J(x) equals:

$$J(x) = \sum \limits _{i=1} ^{m} || x_{i} - C || ^2$$

If C is the centroid, then:

$$C = \frac {\sum \limits _{i=1} ^{m} \ x_{i}}{ m}$$

(Jeffares, p. 1).

B3. Advantages/Benefit of the Tool

Tools will be used:

For this assessment, I'll use Python because the study will be supported by Jupyter notebooks in Python and I Python. Python includes many established data science and machine learning tools, , straightforward, and extensible programming style, and grammar. Python is cross-platform, so it will function whether the analysis is viewed on a Windows PC or a MacBook laptop. When compared to other programming languages such as R or MATLAB, it is quick (Massaron, p. 8[4])[5]. In addition, Python is often regarded in popular media as the most widely used programming language for data science and media (CBTNuggets, p. 1). [6]

NumPy used to work with arrays,
Pandas used to load datasets,
Matplotlib used to plot charts,
Scikit-learn used for machine learning model classes,
SciPy used for mathematical problems, specifically linear algebra transformations, and
Seaborn used for a high-level interface and appealing visualizations.
Using the Pandas library and its accompanying "read csv" function to transform our data as a dataframe is a quick, exact example of loading a dataset and constructing a variable efficiently: imported pandas as pd, df(dataframe) = pd.read csv('ChurnData.csv')

PART III. Data Objectives:

C1. The following will be part of Preparation of data:

We must assess the entire dataset that is free of anomalies as a crucial preprocessing goal. It's also crucial for our meaningful k-means clustering that we determine whether those independent binary variables should be encoded as dummy variables so that they can be included in our studies. For with this unsupervised "classification" method, we ultimately decided to exclude binary dummy variables. In our scatter plots, some variables, such as the critical Churn variable and the significant Age variable, showed uniform distributions. The commented-out code, on the other hand, has been saved for future reference.

C2. Variables in the Dataset:

The variables in the original data that were considered to do the analysis are listed below and classed as continuous or categorical.

Except for the four columns of identification numbers at the beginning of the csv, the grid of features (columns we wish to maintain to find patterns) comprises all features.

Those will be taken out during the cleaning.

Tenure, Churn, Bandwidth GB Year, and MonthlyCharge will be used for visualization reasons.

Continuous	**Categorical**
Children	Techie
Age	Contract
Income	Port_modem
Outage_sec_perweek	Tablet
Email	InternetService
Contacts	Phone
Yearly_equip_failure	Multiple
Tenure	OnlineSecurity
MonthlyCharge	OnlineBackup
Bandwidth_GB_Year	DeviceProtection
	TechSupport
	StreamingTV
	StreamingMovies

C3. Data Preparation Procedures:

- Using Pandas' read csv command, read the data collection into python programming.
- Using the info () and description() methods, evaluate the data for a better understanding of the input data.
- Using the variable "churn df" to name the dataset, and "df" to name the data frame's subsequent usable slices.
- Check for misspellings, strange variable names, and data that is missing.
- Identify outliers that may create or obscure statistical significance using histograms, Scatter plots and box plots.
- Computing replaces missing data with relevant central tendency measures (mean, median, or mode) or just Outliers a few standard deviations above the mean are removed.
- Remove non-essential categorical variables from the dataset to create a purely numerical dataframe for analysis.
- For usage in the K-means clustering model, save the cleaned dataset as "churn prepared kmeans.csv."

The dependent variable "Churn," which is binary and categorical and has values, "Yes" or "No," is extremely crucial to our decision-making process. Our categorical target variable, or y, will be "churn."

The following consistent explanatory factors may be relevant after cleaning the data:

- Bandwidth_GB_Year
- MonthlyCharge
- Tenure (the length of time a customer has been with the company)
- Yearly_equip_failure
- Contacts
- Email
- Outage_sec_perweek
- Income
- Age
- Children

Similarly, the based on this background factors' relevance may be discovered (with only two values, "Yes" or "No" all binary categorical variables, except where noted). The following values will be encoded as 1/0 dummy variables:

- StreamingMovies: Is the customer able to access on-demand movies (yes, no)
- StreamingTV: Whether the consumer has access to streaming television (yes, no)
- TechSupport: Is there a technical assistance add-on for the customer? (No, yes)
- DeviceProtection: Is the consumer eligible for a device protection add-on? (no,yes)
- OnlineBackup: Whether the consumer has purchased an add-on for internet backup (yes, no)
- OnlineSecurity: Whether the consumer has an online security add-on? (no, yes)
- Multiple: Whether or whether the consumer has more than one line of credit (yes, no)
- Phone: Whether the customer has access to a phone line (yes, no)
- InternetService: Customer's internet service provider fiber optic, None,DSL)
- Whether or not the customer possesses a tablet, Surface or iPad (no,yes)
- Whether the customer has a portable modem is determined by port modem (yes, no)
- Customer contract: contract terms of customer (one year, month-to-month, two year)
- Techie: Whether the customer perceives themselves to be technically savvy (as determined by a customer questionnaire completed when they enrolled for services) (no,yes)

In the decisionmaking process, discrete ordinal predictor variables created from consumer surve y responses about various customer service attributes could be valuable. Customers in the surve ys eight customer service rated based on ordinal numerical data aspects on a scale of 8 to 1 (8 being the most essential and 1 being the least important):

- Active_listening - Item1
- Courteous_exchange - Item2
- Respectful_response - Item3
- Options - Item4
- Reliability - Item5
- Timely_replacements - Item6
- Timely_fixes - Item7
- Timely_responses - Item8

1. **Include standard imports all the required references:**

```
In [1]: # Standard data science imports
        import numpy as np
        import pandas as pd
        from pandas import Series, DataFrame

        # Visualization libraries
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline

        # Scikit-learn
        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
```

2. **Change font and color of the Matplotlib:**

```
In [2]: # Change color of Matplotlib font
        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
```

3. **Increase display cell-width**

```
In [3]: # Increase Jupyter display cell-width
        from IPython.core.display import display, HTML
        display(HTML("<style>.container { width:75% !important; }</style>"))
```

4. **Ignore warning codes**

```
In [4]: # Ignore Warning Code
        import warnings
        warnings.filterwarnings('ignore')
```

5. Dataset

```
In [5]: # Load data set into Pandas dataframe
        churn_df = pd.read_csv('C:/Kailash/Rekha/D212/data/churn_clean.csv')
```

6. Dataset size

```
In [6]: # Get an idea of dataset size
        churn_df.shape

Out[6]: (10000, 50)
```

7. Data set features

```
In [7]: # Examine the features of the dataset
        churn_df.columns

Out[7]: Index(['CaseOrder', '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')
```

8. Data frame Info

In [8]:
```
# View DataFrame info
churn_df.info
```

Out[8]:
```
<bound method DataFrame.info of       CaseOrder Customer_id                           Interaction  \
0             1    K409198  aa90260b-4141-4a24-8e36-b04ce1f4f77b
1             2    S120509  fb76459f-c047-4a9d-8af9-e0f7d4ac2524
2             3    K191035  344d114c-3736-4be5-98f7-c72c281e2d35
3             4     D90850  abfa2b40-2d43-4994-b15a-989b8c79e311
4             5    K662701  68a861fd-0d20-4e51-a587-8a90407ee574
...         ...        ...                                   ...
9995       9996    M324793  45deb5a2-ae04-4518-bf0b-c82db8dbe4a4
9996       9997    D861732  6e96b921-0c09-4993-bbda-a1ac6411061a
9997       9998    I243405  e8307ddf-9a01-4fff-bc59-4742e03fd24f
9998       9999    I641617  3775ccfc-0052-4107-81ae-9657f81ecdf3
9999      10000     T38070  9de5fb6e-bd33-4995-aec8-f01d0172a499

                                   UID         City State  \
0     e885b299883d4f9fb18e39c75155d990  Point Baker    AK
1     f2de8bef964785f41a2959829830fb8a  West Branch    MI
2     f1784cfa9f6d92ae816197eb175d3c71      Yamhill    OR
3     dc8a365077241bb5cd5ccd305136b05e      Del Mar    CA
4     aabb64a116e83fdc4befc1fbab1663f9    Needville    TX
...                                ...          ...   ...
9995  9499fb4de537af195d16d046b79fd20a  Mount Holly    VT
9996  c09a841117fa81b5c8e19afec2760104  Clarksville    TN
9997  9c41f212d1e04dca84445019bbc9b41c     Mobeetie    TX
9998  3e1f269b40c235a1038863ecf6b7a0df   Carrollton    GA
9999  0ea683a03a3cd544aefe8388aab16176 Clarkesville    GA

                    County    Zip       Lat        Lng  ...  MonthlyCharge  \
0     Prince of Wales-Hyder  99927  56.25100 -133.37571  ...     172.455519
1                   Ogemaw  48661  44.32893  -84.24080  ...     242.632554
2                  Yamhill  97148  45.35589 -123.24657  ...     159.947583
3                San Diego  92014  32.96687 -117.24798  ...     119.956840
4                Fort Bend  77461  29.38012  -95.80673  ...     149.948316
...                    ...    ...       ...        ...  ...            ...
9995               Rutland   5758  43.43391  -72.78734  ...     159.979400
9996            Montgomery  37042  36.56907  -87.41694  ...     207.481100
9997               Wheeler  79061  35.52039 -100.44180  ...     169.974100
9998               Carroll  30117  33.58016  -85.13241  ...     252.624000
9999             Habersham  30523  34.70783  -83.53648  ...     217.484000

     Bandwidth_GB_Year Item1 Item2  Item3  Item4  Item5 Item6 Item7 Item8
0           904.536110     5     5      5      3      4     4     3     4
1           800.982766     3     4      3      3      4     3     4     4
2          2054.706961     4     4      2      4      4     3     3     3
3          2164.579412     4     4      4      2      5     4     3     3
4           271.493436     4     4      4      3      4     4     4     5
...                ...   ...   ...    ...    ...    ...   ...   ...   ...
9995       6511.252601     3     2      3      3      4     3     2     3
9996       5695.951810     4     5      5      4      4     5     2     5
9997       4159.305799     4     4      4      4      4     4     4     5
9998       6468.456752     4     4      6      4      3     3     5     4
9999       5857.586167     2     2      3      3      3     3     4     1

[10000 rows x 50 columns]>
```

9. Data types

In [9]:
```
# Get data types of features
churn_df.dtypes
```

Out[9]:
```
CaseOrder                 int64
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
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
```

10. Data set

In [10]:
```
# Examine first few records of dataset
churn_df.head()
```

Out[10]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	...	MonthlyCharge
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	...	172.455519
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	...	242.632554
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	...	159.947583
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.24798	...	119.956840
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.80673	...	149.948316

5 rows × 50 columns

11. Descriptive statics

In [11]:
```
# Get an overview of descriptive statistics
churn_df.describe()
```

Out[11]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	Outage_sec_perweek	Email	...	MonthlyCharge	Bandwidth_GB_Year
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.000000	10000.000000	10000.000000	10000.000000	...	10000.000000	10000.000000
mean	5000.50000	49153.319600	38.757567	-90.782536	9756.562400	2.0877	53.078400	39806.926771	10.001848	12.016000	...	172.624816	3392.341550
std	2886.89568	27532.196108	5.437389	15.156142	14432.698671	2.1472	20.698882	28199.916702	2.976019	3.025898	...	42.943094	2185.294852
min	1.00000	601.000000	17.966120	-171.688150	0.000000	0.0000	18.000000	348.670000	0.099747	1.000000	...	79.978860	155.506715
25%	2500.75000	26292.500000	35.341828	-97.082812	738.000000	0.0000	35.000000	19224.717500	8.018214	10.000000	...	139.979239	1236.470827
50%	5000.50000	48869.500000	39.395800	-87.918800	2910.500000	1.0000	53.000000	33170.605000	10.018560	12.000000	...	167.484700	3279.536903
75%	7500.25000	71866.500000	42.106908	-80.088745	13168.000000	3.0000	71.000000	53246.170000	11.969485	14.000000	...	200.734725	5586.141370
max	10000.00000	99929.000000	70.640660	-65.667850	111850.000000	10.0000	89.000000	258900.700000	21.207230	23.000000	...	290.160419	7158.981530

8 rows × 23 columns

12. Remove categorical variables from dataset

In [12]:
```
# Remove less relevant categorical variables from dataset
churn_df = churn_df.drop(columns=['CaseOrder', 'Customer_id', 'Interaction', 'UID'])
```

13. **Using pandas read the data from clean data file and change the names of the last eight survey columns to better describe the variables:**

```
# Load data set into Pandas dataframe
churn_df = pd.read_csv("C:/Rekha/churn_clean.csv")

# Rename last 8 survey columns for better description of variables
churn_df.rename(columns = {'Item1':'Timely_Response',
'Item2':'Timely_Fixes',
'Item3':'Timely_Replacements',
'Item4':'Reliability',
'Item5':'Options',
'Item6':'Respectful_Response',
'Item7':'Courteous_exchange',
'Item8':'Active_Listening'},
inplace=True)
```

14. **Churn data frame with values:**

In [14]:
```
# Review changes
churn_df.head()
```

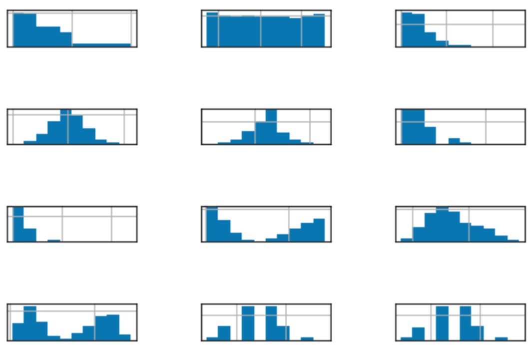
Out[14]:

	City	State	County	Zip	Lat	Lng	Population	Area	TimeZone	Job	...	MonthlyCharge	Bandwidth_GB_Year	Timely_Responses	Timely_Fixes	Timely_Replac
0	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	38	Urban	America/Sitka	Environmental health practitioner	...	172.455519	904.536110	5	5	
1	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	10446	Urban	America/Detroit	Programmer, multimedia	...	242.632554	800.982766	3	4	
2	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	Urban	America/Los_Angeles	Chief Financial Officer	...	159.947583	2054.706961	4	4	
3	Del Mar	CA	San Diego	92014	32.96687	-117.24798	13863	Suburban	America/Los_Angeles	Solicitor	...	119.956840	2164.579412	4	4	
4	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	Suburban	America/Chicago	Medical illustrator	...	149.948316	271.493436	4	4	

5 rows × 46 columns

15. To create histograms:

```
In [16]: # Create histograms of contiuous variables & categorical variables
         churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
                   'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
                   'Bandwidth_GB_Year', 'Timely_Responses', 'Courteous_exchange']].hist()

         plt.tight_layout()
```

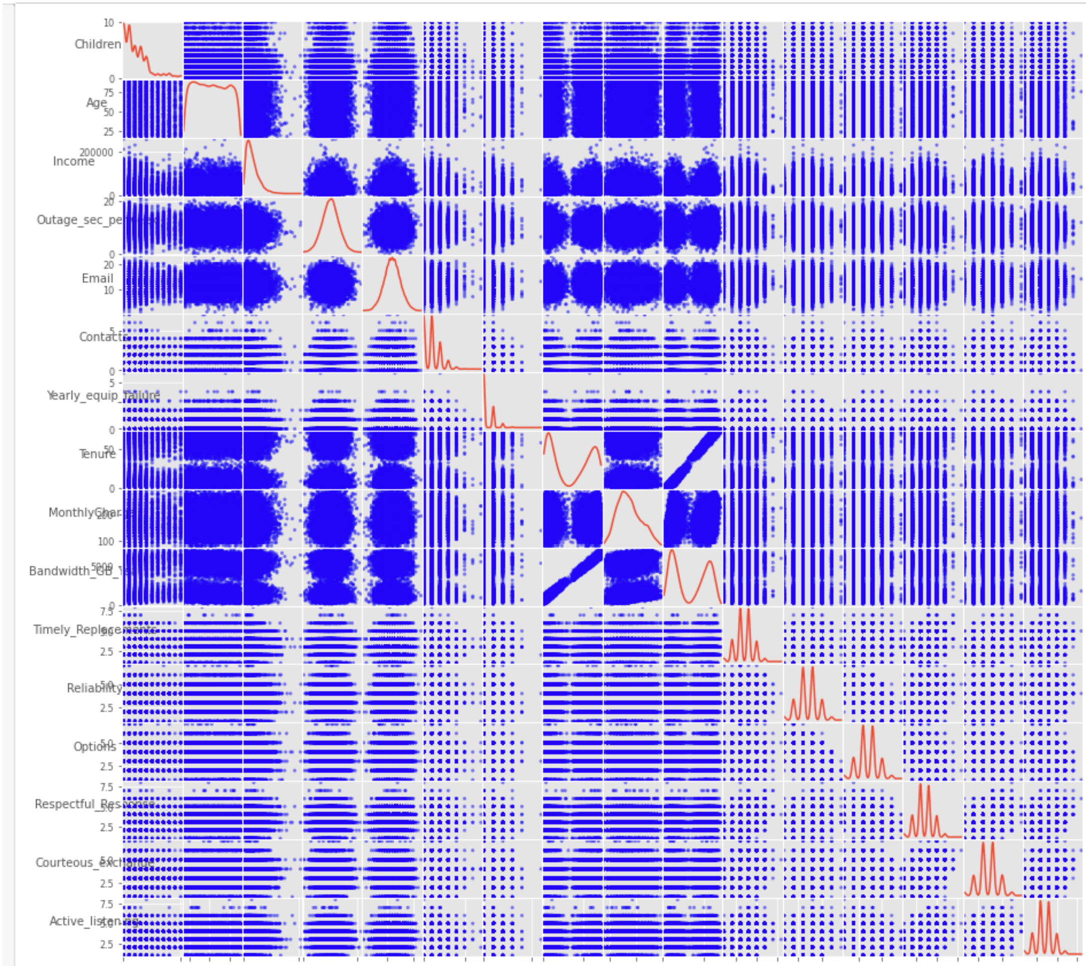


16. To plot style to ggplot:

```
In [17]: # Set plot style to ggplot for aesthetics & R style
         plt.style.use('ggplot')
```

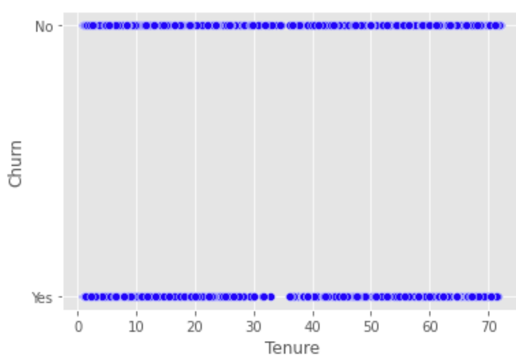
17. List the high-level overview of potential relationships and distributions:

```
In [18]: # Provide a scatter matrix of numeric variables for high level overview of potential relationships & distributions
         churn_numeric = churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek',
                                   'Email', 'Contacts','Yearly_equip_failure', 'Tenure',
                                   'MonthlyCharge', 'Bandwidth_GB_Year', 'Timely_Replacements',
                                   'Reliability', 'Options', 'Respectful_Response', 'Courteous_exchange',
                                   'Active_listening']]


         scatter_matrix = pd.plotting.scatter_matrix(
             churn_numeric,
             figsize  = [15, 15],
             diagonal = "kde",
             color="b"
         )

         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)
```



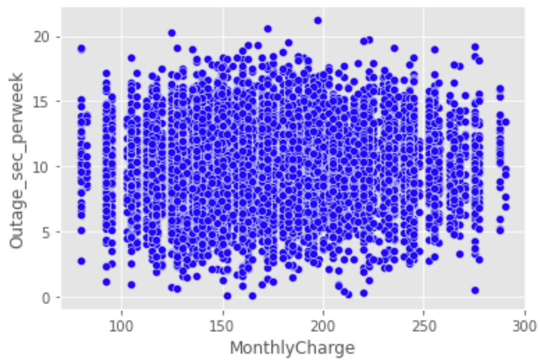
18. scatterplot of Tenure:

In [19]:
```
# Create a scatterplot to get an idea of correlations between potentially related variables
sns.scatterplot(x=churn_df['Tenure'], y=churn_df['Churn'], color='blue')
plt.show();
```



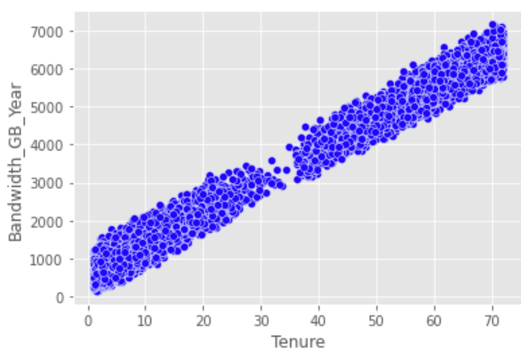
19. scatterplot of Monthly charge:

In [20]:
```
# Create a scatterplot to get an idea of correlations between potentially related variables
sns.scatterplot(x=churn_df['MonthlyCharge'], y=churn_df['Outage_sec_perweek'], color='blue')
plt.show();
```



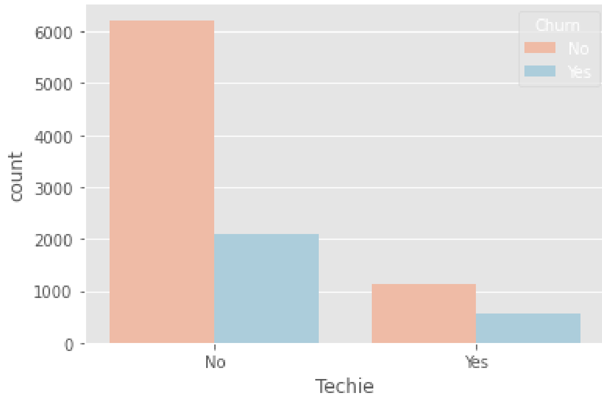
20. scatterplot of Tenure and Bandwidth_GB_Year:

In [21]:
```
# Create a scatterplot to get an idea of correlations between potentially related variables
sns.scatterplot(x=churn_df['Tenure'], y=churn_df['Bandwidth_GB_Year'], color='blue')
plt.show();
```



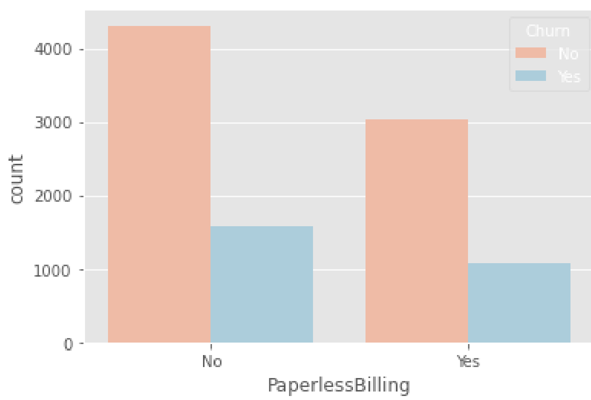
21. scatter_matrix:

In [22]:
```
# Countplot more useful than scatter_matrix when features of dataset are binary
plt.figure()
sns.countplot(x='Techie', hue='Churn', data=churn_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



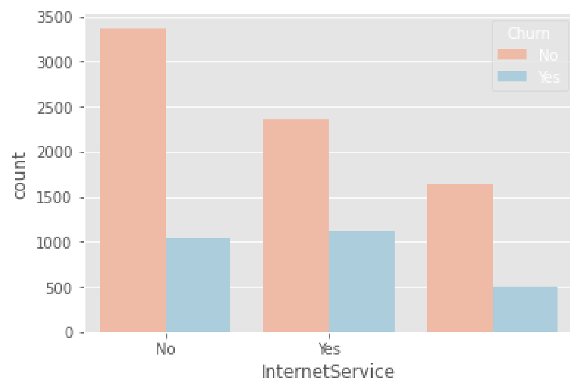
22. scatter matrix paperless Billing

In [23]:
```
# Countplot more useful than scatter_matrix when features of dataset are binary
plt.figure()
sns.countplot(x='PaperlessBilling', hue='Churn', data=churn_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



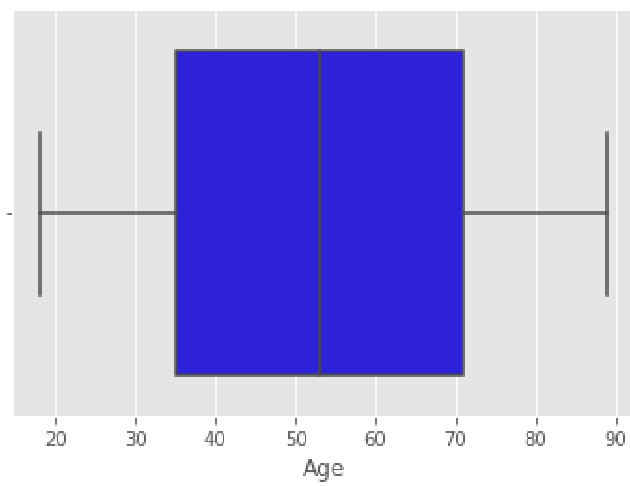
23. scatter matrix Internet Service:

In [24]:
```
# Countplot more useful than scatter_matrix when features of dataset are binary
plt.figure()
sns.countplot(x='InternetService', hue='Churn', data=churn_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



24. Seaborn boxplots for continuous & categorical variables

In [25]:
```
# Create Seaborn boxplots for continuous & categorical variables
sns.boxplot('Age', data = churn_df, color="blue")
plt.show()
```



25. Find exact Age range in column

In [26]:
```
# Find exact Age range in column
print("Minimum Age is", churn_df.Age.min())
print("Maximum Age is", churn_df.Age.max())
print("Age range is", churn_df.Age.max()-churn_df.Age.min())
```

```
Minimum Age is 18
Maximum Age is 89
Age range is 71
```

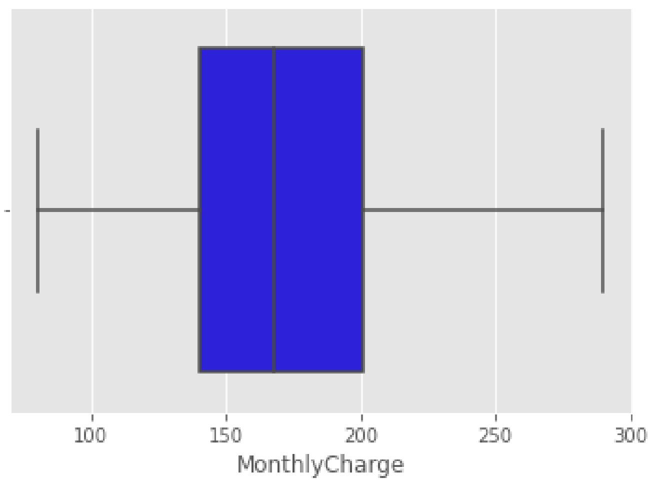
26. Find exact Income range in column

In [27]:
```
# Find exact Income range in column
print("Minimum Income is", int(churn_df.Income.min()))
print("Maximum Income is", int(churn_df.Income.max()))
print("Income range is", int(churn_df.Income.max()-churn_df.Income.min()))
```

```
Minimum Income is 348
Maximum Income is 258900
Income range is 258552
```

27. Create Seaborn boxplots for Monthly Charge:

In [28]:
```
# Create Seaborn boxplots for continuous & categorical variables
sns.boxplot('MonthlyCharge', data = churn_df, color="blue")
plt.show()
```



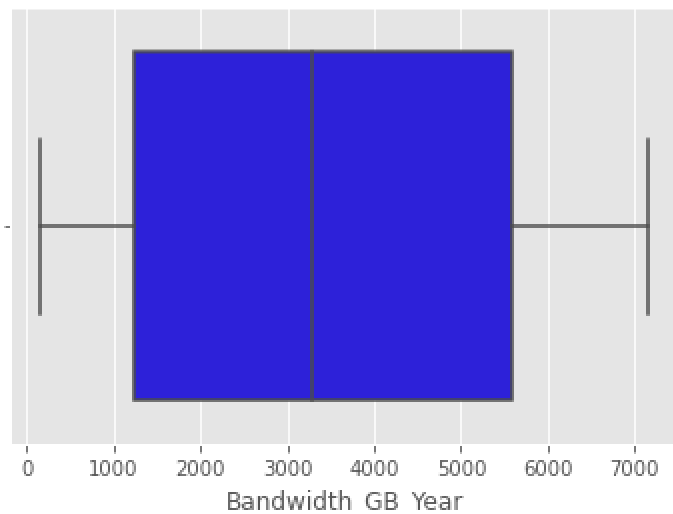
28. Find exact MonthlyCharge range in column

In [29]:
```
# Find exact MonthlyCharge range in column
print("Minimum MonthlyCharge is", int(churn_df.MonthlyCharge.min()))
print("Maximum MonthlyCharge is", int(churn_df.MonthlyCharge.max()))
print("MonthlyCharge range is", int(churn_df.MonthlyCharge.max()-churn_df.MonthlyCharge.min()))
```

```
Minimum MonthlyCharge is 79
Maximum MonthlyCharge is 290
MonthlyCharge range is 210
```

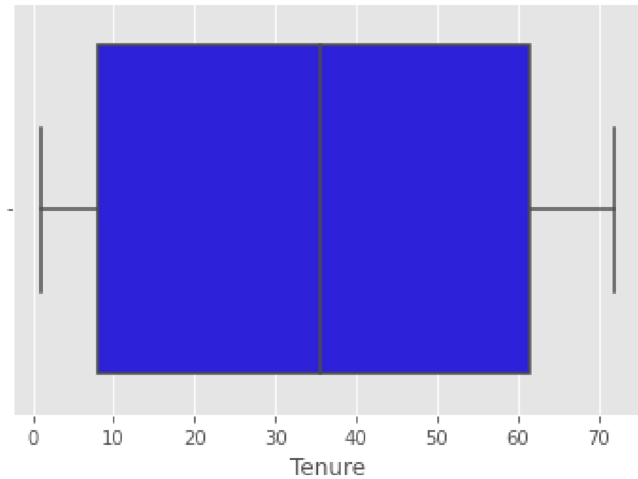
29. Create Seaborn boxplots for Bandwidth_GB_Year:

In [30]:
```
# Create Seaborn boxplots for continuous & categorical variables
sns.boxplot('Bandwidth_GB_Year', data = churn_df, color="blue")
plt.show()
```



30. Seaborn boxplots for tenure

```
In [31]: # Create Seaborn boxplots for continuous variables
         sns.boxplot('Tenure', data = churn_df, color="blue")
         plt.show()
```



31. missing data points within dataset

In [32]:
```
# Discover missing data points within dataset
data_nulls = churn_df.isnull().sum()
print(data_nulls)
```

```
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
Timely_Responses        0
Timely_Fixes            0
Timely_Replacements     0
Reliability             0
Options                 0
Respectful_Response     0
Courteous_exchange      0
Active_listening        0
dtype: int64
```

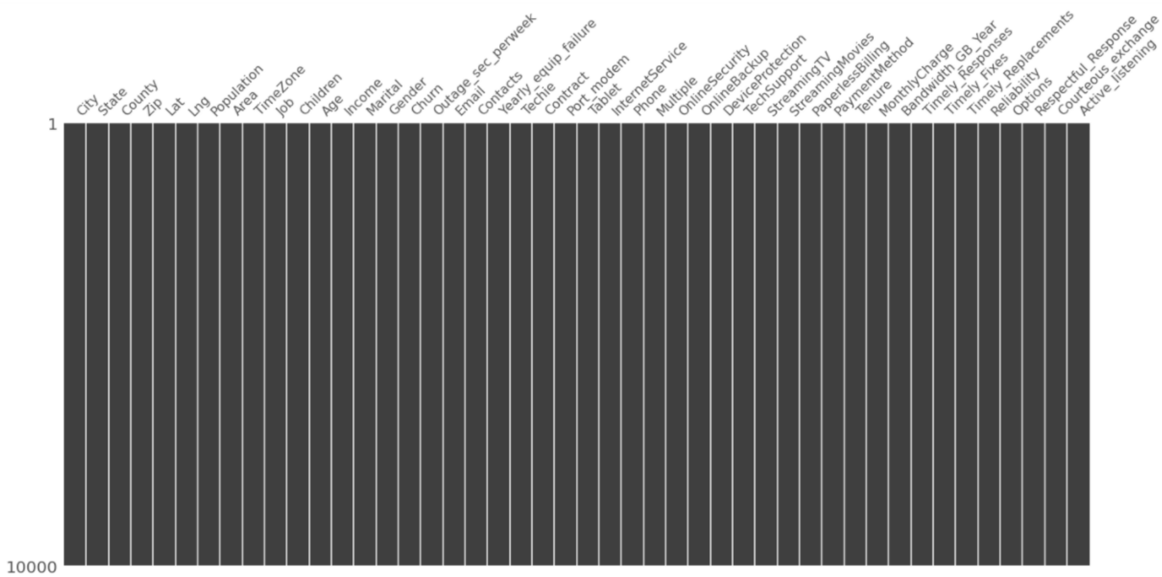
C4: Imagination/Visualization

In [33]:

```
# Check for missing data & visualize missing values in dataset

# Install appropriate library
!pip install missingno

# Importing the libraries
import missingno as msno

# Visualize missing values as a matrix
msno.matrix(churn_df);
"""(GeeksForGeeks, p. 1)"""
```

```
Requirement already satisfied: missingno in c:\users\kaila\anaconda3\lib\site-packages (0.5.0)
Requirement already satisfied: matplotlib in c:\users\kaila\anaconda3\lib\site-packages (from missingno) (3.3.4)
Requirement already satisfied: numpy in c:\users\kaila\anaconda3\lib\site-packages (from missingno) (1.20.1)
Requirement already satisfied: seaborn in c:\users\kaila\anaconda3\lib\site-packages (from missingno) (0.11.1)
Requirement already satisfied: scipy in c:\users\kaila\anaconda3\lib\site-packages (from missingno) (1.6.2)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (8.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\kaila\anaconda3\lib\site-packages (from matplotlib->missingno) (1.3.1)
Requirement already satisfied: six in c:\users\kaila\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib->missingno) (1.15.0)
Requirement already satisfied: pandas>=0.23 in c:\users\kaila\anaconda3\lib\site-packages (from seaborn->missingno) (1.2.4)
Requirement already satisfied: pytz>=2017.3 in c:\users\kaila\anaconda3\lib\site-packages (from pandas>=0.23->seaborn->missingno) (2021.1)
```

Out[33]: '(GeeksForGeeks, p. 1)'



In [34]:
```
# # Encode binary categorical variables with dummies
# churn_df['DummyGender'] = [1 if v == 'Male' else 0 for v in churn_df['Gender']]
# churn_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in churn_df['Churn']] ### If the customer left (churned) they get a '1'
# churn_df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in churn_df['Techie']]
# churn_df['DummyContract'] = [1 if v == 'Two Year' else 0 for v in churn_df['Contract']]
# churn_df['DummyPort_modem'] = [1 if v == 'Yes' else 0 for v in churn_df['Port_modem']]
# churn_df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in churn_df['Tablet']]
# churn_df['DummyInternetService'] = [1 if v == 'Fiber Optic' else 0 for v in churn_df['InternetService']]
# churn_df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in churn_df['Phone']]
# churn_df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in churn_df['Multiple']]
# churn_df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in churn_df['OnlineSecurity']]
# churn_df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in churn_df['OnlineBackup']]
# churn_df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in churn_df['DeviceProtection']]
# churn_df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in churn_df['TechSupport']]
# churn_df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in churn_df['StreamingTV']]
# churn_df['StreamingMovies'] = [1 if v == 'Yes' else 0 for v in churn_df['StreamingMovies']]
# churn_df['DummyPaperlessBilling'] = [1 if v == 'Yes' else 0 for v in churn_df['PaperlessBilling']]
```

In [35]:
```
# # # Drop original categorical 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'])
```

In [36]:
```
# List features for analysis
features = (list(churn_df.columns[:-1]))
print('Features for analysis include: \n', features)
```

```
Features for analysis include:
 ['City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Children', 'Age', 'Income', 'Marital', 'Gender', 'Churn', 'Outage_sec_per
week', 'Email', 'Contacts', 'Yearly_equip_failure', 'Techie', 'Contract', 'Port_modem', 'Tablet', 'InternetService', 'Phone', 'Multiple', 'OnlineSecurity', 'OnlineBa
ckup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'PaymentMethod', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Tim
ely_Responses', 'Timely_Fixes', 'Timely_Replacements', 'Reliability', 'Options', 'Respectful_Response', 'Courteous_exchange']
```

C4: Dataset that has been cleaned:

"churn_prepared_kmeans.csv" cleaned data set.

churn_prepared_kmeans.csv

In [38]:
```
# Extract Clean dataset
churn_df.to_csv('C:/Kailash/Rekha/D212/data/churn_prepared_kmeans.csv')
```

PART IV. Analysis/Research

D1. Calculations for the Output and Intermediate:

We iterated over 3 pairs of matched feature sets through using Scikit-learn KMeans class to generate meaningful heuristics from which to better understand our consumers and, as a result, make better business decisions. Among the three sets were:

- Tenure and Monthly Charge
    - Object includes 6 clusters of customers

```
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')
```

- Monthly Charge and Income
    - Object includes 4 clusters of customers

```
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')
```

- Bandwidth_GB_Year and Tenure
    - Object includes 2 clusters of customers

```
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')
```

For each set, we applied the slide plot elbow technique to determine the ideal number of clusters.

D2. Execution of Code:

Below are the K-means clustering code and graphics.

In [39]:
```
# Import prepared Churn dataset
churn_df = pd.read_csv('C:/Kailash/Rekha/D212/data/churn_prepared_kmeans.csv', index_col=0)
```

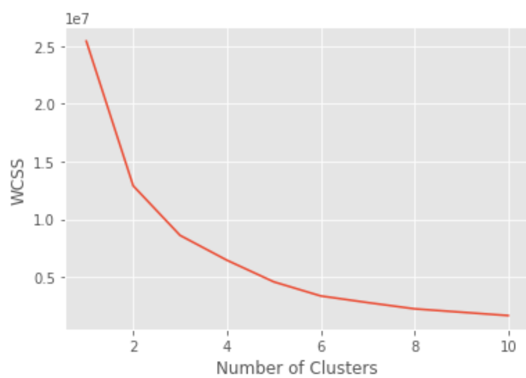
In [40]:
```
# Import KMeans class from Scikit-Learn
from sklearn.cluster import KMeans
```

In [41]:
```
# Set plot style to ggplot for aesthetics & R style
plt.style.use('ggplot')
```

K-means: Tenure v. MonthlyCharge

In [42]:
```
# Select indexes (features) of Tenure and MonthlyCharge for initial clustering
X = churn_df.iloc[:, [35, 36]].values
```

In [43]:
```
# Use the elbow method to find the optimal number of clusters
# Create a Within Cluster Sum of Squares (WCSS) list
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('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.savefig('churn_scree_tenure_v_monthly-charge.jpg')
plt.show()
```



In [44]:
```python
# Train the K-means model on the dataset
kmeans = KMeans(n_clusters=6, init='k-means++', random_state=42)

# Build the dependent variable to split customers in different clusters
y_kmeans = kmeans.fit_predict(X)
```

In [45]:
```python
print(y_kmeans)
```

```
[4 3 4 ... 1 5 5]
```

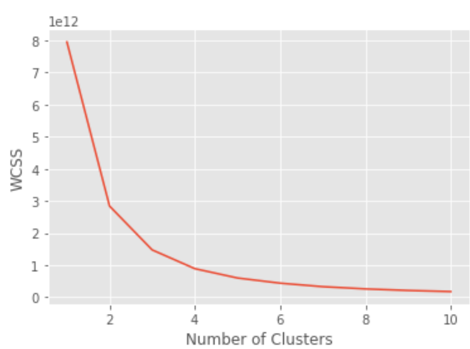
In [46]:
```python
# Visualize the clusters
# Scatter plot 5 clusters for Tenure v. MonthlyCharge
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 10, c = 'red', label = 'cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 10, c = 'green', label = 'cluster 2')
plt.scatter(X[y_kmeans == 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 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 Clusters of Customers') #get the title property handler
plt.getp(title_obj)                    #print out the properties of title
plt.getp(title_obj, 'text')            #print out the 'text' property for title
plt.setp(title_obj, color='gray')         #set the color of title to red

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')

# Save plot to directory
plt.savefig('churn_kmeans_tenure_v_monthly-charge.jpg')

# 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 = white
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 Clusters of Customers
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
```

K-means: Income v. MonthlyCharge

In [47]:
```
# Select indexes (features) of Income and MonthlyCharge for initial clustering
X = churn_df.iloc[:, [12, 36]].values
```

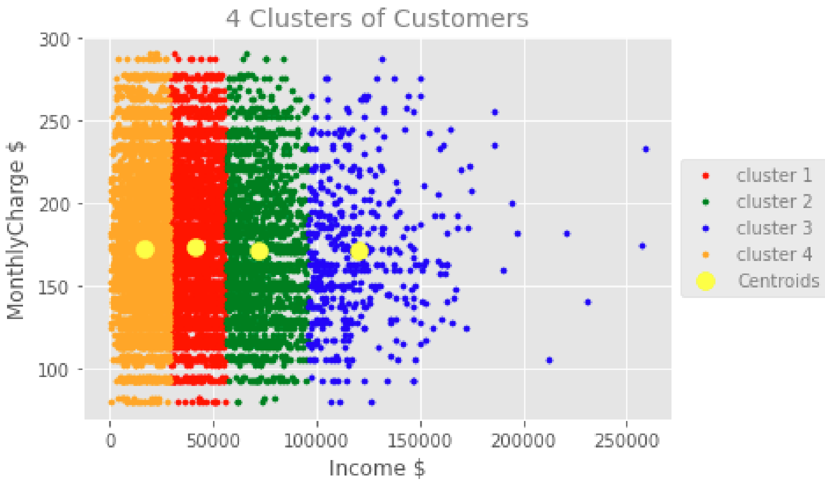
In [48]:
```
# Use the elbow method to find the optimal number of clusters
# Create a Within Cluster Sum of Squares (WCSS) list
wcss = []

# 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('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.savefig('churn_scree_income_v_monthly-charge.jpg')
plt.show()
```



In [49]:
```
# Train the K-means model on the dataset
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)

# Build the dependent variable to split customers in different clusters
y_kmeans = kmeans.fit_predict(X)
```

In [50]:
```
# Visualize the clusters
# Scatter plot 4 clusters for Income and 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 Clusters of Customers') #get the title property handler
plt.getp(title_obj)                    #print out the properties of title
plt.getp(title_obj, 'text')            #print out the 'text' property for title
plt.setp(title_obj, color='gray')         #set the color of title to red

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')

# Save plot to directory
plt.savefig('churn_kmeans_income_v_monthly-charge.jpg')

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

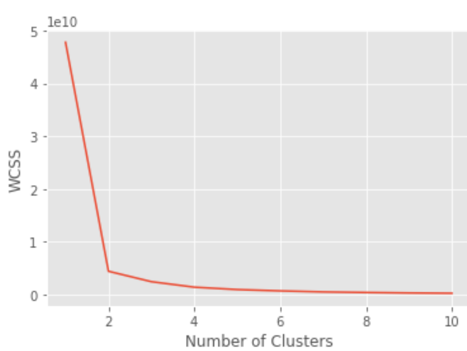
```
agg_filter = None
alpha = None
animated = False
bbox_patch = None
children = []
clip_box = None
clip_on = True
clip_path = None
color or c = white
contains = None
figure = Figure(432x288)
fontfamily or family = ['sans-serif']
fontname or name = DejaVu Sans
fontproperties or font or font_properties = 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 Clusters of Customers
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
```



K-means: Tenure v. Bandwidth_GB_Year

In [51]:
```
# Select indexes (features) of Tenure and Bandwidth_GB_Year for initial clustering
X = churn_df.iloc[:, [35, 37]].values
```

In [52]:
```
# Use the elbow method to find the optimal number of clusters
# Create a Within Cluster Sum of Squares (WCSS) list
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('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.savefig('churn_scree_tenure_v_bandwidth-gb-year.jpg')
plt.show()
```



In [53]:
```python
# Train the K-means model on the dataset
kmeans = KMeans(n_clusters=2, init='k-means++', random_state=42)

# Build the dependent variable to split customers in different clusters
y_kmeans = kmeans.fit_predict(X)
```

In [54]:
```python
# Visualize the clusters
# Scatter plot 4 clusters for Tenure and 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 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 Clusters of Customers') #get the title property handler
plt.getp(title_obj)                    #print out the properties of title
plt.getp(title_obj, 'text')            #print out the 'text' property for title
plt.setp(title_obj, color='gray')         #set the color of title to red

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')

# Save plot to directory
plt.savefig('churn_kmeans_tenure_v_bandwidth-gb-year.jpg')

# 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 = white
    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 Clusters of Customers
    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. Do the following to describe your data analysis:

E1. Clustering Technique Accuracy

"Validating the clustering technique is somewhat tricky compared to supervised machine learning algorithm since there is no ground truth labels in clustering Procedure," Manimaran writes on TowardsDataScience.com (Manimaran, p. 1).

So, when measuring the accuracy of our k-means clustering, we'll take three criteria into account:

- Number of clusters
- Clustering quality
- Clustering tendency

Number of clusters

We utilized the elbow technique to determine the best number of clusters k by plotting the k values against within-cluster variation, as shown in the scree plots above. Two, four, and six are clustered in our elbow results.

Clustering quality

We can see how tight clusters are in relation to their respective centroids after clustering. Our clusters are not firmly grouped around their centroids, as evidenced by the depicted clusters for our three kmeans clusterings. Instead, we have "levels" or "bands" of clusters due to the nature of the customer dataset.

Clustering tendency

As seen by our scatter matrix above (see bivariate plots including customer survey findings - Replacements, Reliability, etc.), many of our prospective numerical variables contain evenly distributed data points. Our studies will be based on Tenure, non-uniform distributions of, Bandwidth GB Year, MonthlyCharge and Income. As a result, meaningful clusters may be more likely to emerge. We also don't use dummy variables, which are also equally distributed.

E2. Conclusions & Implications:

We must return to our original study question, "Can we better understand our consumers and find patterns specific to consumers who churn utilizing unsupervised learning data mining?" for answers and ramifications.

We employed the Within Cluster Sum of Squares (WCSS), sometimes known as the "elbow" method, to find the best clustering algorithm k for our three bivariate clustering's. The following are the findings and their implications:

- We observed two primary categories when we compared bandwidth usage yearly to customer tenure with the telecom firm. Customers who stay for a small period and use less GBs, and those who stay for a longer period and use more GBs. This conclusion appears to be self-evident, and it implies that we should try to keep clients.

- We observed four key categories, or perhaps market sectors, when comparing monthly fee to client income. Although we should expect monthly prices to rise in tandem with user income. We couldn't locate this information. Customers' monthly charges ranged from low to high within each customer income cluster We'd want to see higher-income individuals overspend or, at the very least, create marketing tactics that encourage them to spend their disposable cash with us rather than elsewhere.

- Eventually, when monthly charges were compared to customer tenure with the telecom provider, the WCSS recommended six ideal clustering's, which mirrored the preceding results with bandwidth only. This outcome may provide us the best insight of which groups to market to more aggressively, those who pay a lesser monthly fee but stay for longer periods of time, and, hopefully, reduces spam to those who have over spent money with us but are not staying for longer periods of time.

Finally, churn, or short stay with a company, appears to be linked to the use of fewer services and, possibly, spending fewer dollars with us.

E3. Restrictions

The data for this telecom firm dataset does not come from a warehouse, which is a drawback of this investigation. It's as if we used Python statistical libraries to generate the data at random in this instance. As a result, we are unable to contact the personnel that arranged and acquired this data to inquire as to why certain uniformities occur, and whether A/B testing or other comparisons are more relevant to answering issues about customer retention or churn, in their subject-matter-expert judgments. In a real-world project, we'd go to the department where the data was collected and, hopefully, discover more significant results through a more rigorous, focused procedure.

E4: Plan of Action

Marketers and decision-makers should be aware of this our bivariate research suggests certain links. We should look at the qualities that are shared by those who are leaving the company and attempt to reduce the likelihood that they will occur with any future consumer. Early descriptive statistical analyses imply that customers are less likely to abandon a company if they subscribe to more services, such as an online backup or additional port modem. Clearly, it is in the best interests of the company to give customers with more services and enhance their customer experience by supporting them in knowing all the mobile phone service, but a variety of other services are available to them as a subscriber.

Having said that, there is a subset of consumers that earn a lot of money but pay a low monthly fee. More marketing and direct contact from our advertisers should be directed towards these demographics.

There are also pockets of low-income users, with both high and low monthly fees. As an ethical company, we should avoid targeting these market segments because 1) they clearly do not have the financial means to invest on "luxury" services like streaming videos, 2) these customers may be unable to make their monthly payments and may leave our company, migrating to other companies' "free trial" offers, leaving us with an unpaid bill.

PART VI. Documentary Evidence

F. Panopto recording:

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=346a3e96-f762-424f-99c4-ae58011f984c

G. Third Party Evidence:

Title: (Visualize missing values (NaN) values using Missingno Library | Python |), GeeksForGeeks.
Date: July 4th, 2019
URL: https://www.geeksforgeeks.org/python-visualize-missing-values-nan-values-using-missingno-library/

Title: (Machine Learning A-Z: Hands-On Python & R in Data Science), SuperDataScience
Date: August 15th, 2021
URL: https://www.superdatascience.com/

H. References:

1 Author: Jeffares, A.
Date: November 19th,2019.
Title: K-means: A Complete Introduction. TowardDataScience
https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

[2] Author: Daityari, S.
Date: October 3rd,2021
Title: Basics of k-means clustering. DataCamp
https://campus.datacamp.com/courses/cluster-analysis-in-python/k-means-clustering-3?ex=1

[3] Author: VanderPlas, J.
Date: 2017
Title: Python Data Science Handbook. O'Reilly.

[5] Author: Massaron, L. & Boschetti, A.
Date: 2016
Title: Regression Analysis with Python. Packt Publishing.

[6] Author: CBTNuggets
Date: September 20th,2018.
Title: Why Data Scientists Love Python
https://www.cbtnuggets.com/blog/technology/data/why-data-scientists-love-python