WGU D206 PA

Andrew Shrestha

Part I: Research Question

A. Question: Are we able to identify and quantify the risk at which customers will churn on their service provider?

B. Describing Variables in the Data Set

This data set grants us information on 10,000 customers with 50 individual variables pertaining to their personal information and product behavioural habits.

The focus of our analysis is that of the dependent variable "Churn". This column in the data set will give us insights into whether the customer has chosen to continued with the services or chosen to discontinue. The data for this column will be in the form of a boolean (Yes/No) outcome.

The independent variables correlating to the churn outcome in question will be as follows:

- **i.** Services the customer signed up for (Phone, Multiple lines, Internet, Online security, Online backup device protection, Technical support, and Streaming TV and Movies). The data corresponding to these variables will be categorical for the "Internet" column and boolean (yes/no) for the others.
- **ii.** Customer Account Information (Tenure, Contracts, Payment methods, Paperless billing, Monthly charges, GB usage over the year, etc). The data corresponding to these variables will be a mix of categorial and numeric. "Tenure", "Monthly charge", "GB usage over the year" will be Numeric while "Contracts, "Payment methods" and "Paperless billing" will be categorical in this case.
- **iii.** Customer Demographics (Gender, Age, Job, Income, etc). The data corresponding to these variables will be a mix of categorical and numeric. "Gender" and "Job" will be Categorical while "Age" and "Income" will be Numeric in this case.
- **iv.** 8 Survey Questions taken from the customers about various factors on a numeric 1-8 scale. The data corresponding to each of these answers will be in Numeric form.

Part II: Data-Cleaning Plan

A. Creating a Plan to Identify Anomalies

i. Utilizing the python programming language to upload the Churn_Raw_Data CSV file and creating a churn dataframe in which to work with
ii. Familiarize all the columns and rows to get a surface level understanding of the relations between the data and the desired dependent “Churn” variable
iii. Address any unnecessary data where we look at repetitive, irrelevant, or duplicated data. This is to be removed as keeping these values does not add any real value to our business question.
iv. Address any inconsistent data such as upper or lower case variable names/ output for categorical variables, formats, and spelling
v. Address any missing data via creation percentages to get a clearer observation of how many missing data that are present in the data set. Then choose to either drop the observation, impute, or replace the missing information.
vi. Finally, address any clear outliers in the data set descriptive statistics for any numeric outliers, while utilizing bar charts for categorical outliers. Then with this made clear, choose either to drop, correct, or keep the outliers so that the significance of the data remains intact

B. Justifying Approach

Familiarizing myself with the columns and having a surface overview of the data and their relationships with each other and patterns should be the first step since it gives a clear picture and direction for the cleaning process. This will also involve making the data more presentable and of more relevance. We drop any values that may be repeating, irrelevant, or duplicate so that the data can be more compressed into only observations with value to our business question. we want to make sure that everything makes sense and is consistent in terms of punctuation, spelling, grammar, etc… this just makes things easier to interpret and avoids confusion within the data set. This ensures that the data is prepped and ready to go before manipulating the data structure via addressing missing values and outliers.

The characteristics from this data through just basic observations seems to have missing values present in specific columns rather than at random, where Children, Age, Income, and bandwith are often the culprit of these missing values.

This justifies the reasoning of addressing the missing values after the initial observations of patterns in the data set. We want to be dealing with complete values before anything else so this step is of utmost important and should be dealt with first before any other transformations/ interpretations of the data set.

After this step, outliers should be handled because this tends to skew data and results in possibly inaccurate outcomes when taken in together with the other standardized data. It is important to have an explanation as to how the outliers came about and if it is indeed accurate. This justifies the reasoning of addressing outliers second since once you have the full data set, it is important to have accurate data that is true to the variables **Lianne & Justin (2020).**

C. Justifying Programming Language

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 **DataCamp (2020).** 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. This will be 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 **Seaborn (2021).**

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

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

D. Codes for Identifying Anomalies

```
#import the necessary Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt

```

```
#read the data csv and create a dataframe to work with
df = pd.read_csv (r"C:\Users\andre\OneDrive\Desktop\churn_raw_data.csv")
df.drop("Unnamed: 0",axis=1)
df.describe()
```

```
#get surface level information about the data set by looking at the data
structures
print(df.shape)
print(df.dtypes)
df.drop("Unnamed: 0",axis=1)
```

```
# drop irrelevant/less significant variables
df_New = df.drop (columns=['CaseOrder', 'Zip', 'Lat', 'Lng',
'Population','Interaction'])
df_New
```

```
#Addressing any inconsistencies/ typos with the categorical variables
df_New['Job'].unique()
df_New['Area'].unique()
df_New['Timezone'].unique()
df_New['Education'].unique()
df_New['Employment'].unique()
df_New['Marital'].unique()
df_New['Gender'].unique()
df_New['Contract'].unique()
df_New['PaymentMethod'].unique()
df_New['InternetService'].unique()
df_New['City'].unique()
df_New['State'].unique()
df_New['County'].unique()
```

```
#Addressing any duplicated data for variables Numerical variables: Age,
Income, Tenure, MonthlyCharge, Bandwidth_GB_Year

df_New_unduplicated_Age = df_New.drop('Age', axis=1). drop_duplicates()
```

```
print(df_New.shape)
print(df_New_unduplicated_Age.shape)

df_New_unduplicated_Income = df_New.drop('Income', axis=1). drop_duplicates()
print(df_New.shape)
print(df_New_unduplicated_Income.shape)

df_New_unduplicated_Tenure = df_New.drop('Tenure', axis=1). drop_duplicates()
print(df_New.shape)
print(df_New_unduplicated_Tenure.shape)

df_New_unduplicated_MonthlyCharge = df_New.drop('MonthlyCharge', axis=1).
drop_duplicates()
print(df_New.shape)
print(df_New_unduplicated_MonthlyCharge.shape)

df_New_unduplicated_Bandwidth_GB_Year = df_New.drop('Bandwidth_GB_Year',
axis=1). drop_duplicates()
print(df_New.shape)
print(df_New_unduplicated_Bandwidth_GB_Year.shape)
```

```
#Renaming the Customer Responces column to make it clearer what their
numerical responce was in relation to
df_New.rename(columns = {'item1': 'Timely_Response', 'item2':'Timely_Fixes',
'item3':'Timely_Replacements',

'item4':'Reliability','item5':'Options','item6':'Respectful_Response','item7':
'Couteous_Exchange',
                         'item8':'Evidence_Of_Active_Listening'}, inplace=True)
```

```
#Look at how many missing variables there are in the data set from each
variable
data_missing = df_New.isnull().sum()
print(data_missing)
```

```
#take a look at the percentage missing from each variable. This can be used to
decide if we will drop, impute, or replace
#the missing variables
percent_missing = df_New.isnull().sum() * 100 / len(df)
print(percent_missing)
```

```
#Now addressing any outliers on the data set via using 3 Standard Deviation
for the Significant Variables:
#Age, Income, Tenure, Monthly Charge, Bandwidth_GB_Year.

#Finding the upper limit and lower limit for the Significant Variable AGE

upper_limit_Age = df_New.Age.mean() + 3*df_New.Age.std()
upper_limit_Age
```

```
9
10 #Finding the lower limit and lower limit for the Significant Variable AGE
11 lower_limit_Age = df_New.Age.mean() - 3*df_New.Age.std()
12 lower_limit_Age
13
14 #Identifying the outliers in our data frame for AGE
   df_New[(df_New.Age>upper_limit_Age)|(df_New.Age<lower_limit_Age)]
```

```
1 #Finding the upper limit and lower limit for the Significant Variable Income
  upper_limit_Income = df_New.Income.mean() + 3*df_New.Income.std()
2 upper_limit_Income

3 #Finding the upper limit and lower limit for the Significant Variable Income
  lower_limit_Income = df_New.Income.mean() - 3*df_New.Income.std()
4 lower_limit_Income

5 #Identifying the outliers in our data frame for Income
  df_New[(df_New.Income>upper_limit_Income)|(df_New.Income<lower_limit_Income]
6
```

```
  #Finding the upper limit and lower limit for the Significant Variable Tenure
1 upper_limit_Tenure = df_New.Tenure.mean() + 3*df_New.Tenure.std()
  upper_limit_Tenure

2 #Finding the upper limit and lower limit for the Significant Variable Tenure
3 lower_limit_Tenure = df_New.Tenure.mean() - 3*df_New.Tenure.std()
  lower_limit_Tenure

4 #Identifying the outliers in our data frame for Tenure
5 df_New[(df_New.Tenure>upper_limit_Tenure)|(df_New.Tenure<lower_limit_Tenure)]
```

```
1 #Finding the upper limit and lower limit for the Significant Variable
  MonthlyCharge
2 upper_limit_MonthlyCharge = df_New.MonthlyCharge.mean() +
  3*df_New.MonthlyCharge.std()
3 upper_limit_MonthlyCharge

4 #Finding the upper limit and lower limit for the Significant Variable
  MonthlyCharge
5 lower_limit_MonthlyCharge = df_New.MonthlyCharge.mean() -
  3*df_New.MonthlyCharge.std()
6 lower_limit_MonthlyCharge

7 #Identifying the outliers in our data frame for MonthlyCharge
```

```
df_New[(df_New.MonthlyCharge>upper_limit_MonthlyCharge)|(df_New.MonthlyCharge<
lower_limit_MonthlyCharge)]
```

```
#Finding the upper limit and lower limit for the Significant Variable
Bandwidth_GB_Year
upper_limit_Bandwidth_GB_Year = df_New.Bandwidth_GB_Year.mean() +
3*df_New.Bandwidth_GB_Year.std()
upper_limit_Bandwidth_GB_Year

#Finding the upper limit and lower limit for the Significant Variable
Bandwidth_GB_Year
lower_limit_Bandwidth_GB_Year = df_New.Bandwidth_GB_Year.mean() -
3*df_New.Bandwidth_GB_Year.std()
lower_limit_Bandwidth_GB_Year

#Identifying the outliers in our data frame for Bandwidth_GB_Year
df_New[(df_New.Bandwidth_GB_Year>upper_limit_Bandwidth_GB_Year)|(df_New.Bandwi
dth_GB_Year<lower_limit_Bandwidth_GB_Year)]
```

Part III: Data Cleaning

A. Describing Findings of Anomalies from Implementing the Data Cleaning Plan

The Findings in this data set were that there seemed to be no errors in terms of consistency among the categorical variables. There also seemed to be no duplicates among the significant numerical data variables that were needed to be dropped. This was done through the use of unique() and drop_duplicates() functions in python.

There were missing values found for the numerical variables of: Children, Age, Income, Tenure, and Bandwidth_GB_Year. Likewise, missing categorical variables of: Techie, Phone, Techsupport. This was done through the use of isnull().sum().

Of the chosen significant variables of interest for the quantitative data(Age, Income, Tenure, MonthlyCharge, and Bandwidth_GB_Year), there were only outliers present in Income and MonthlyCharge. In this case, the outliers were calculated through values being more than 3 standard deviations away from the mean. This was done by creating upper and lower limits for each variable for 3 standard deviations away from mean. This was also supported visually via boxplots (As seen in the Mitigation Code part).

B. Justification for Mitigating Each type of Anomaly

As stated above, the missing variables found in the data were in the variables: Children, Age, Income, Tenure, Bandwidth_GB_Year, Techie, Phone, and Techsupport. The mitigation method I used for these missing variables first was to convert them into percentages compared to their respective variable totals and see whether or not I would be keeping them or imputing them. Since there were all below the 25% threshold, I decided to not drop the rows associated with the missing values, but instead impute them with the median values to get a more accurate result **Anthony B. Ryder, Anna V. Wilkinson, Michelle**

K. McHugh, Katherine Saunders, Sumesh Kachroo, Anthony D'Amelio, Jr., Melissa Bondy, and Carol J. Etzel (2011).

As for the outliers for the Income and MonthlyCharge variables, I decided to leave them in because the amount of outliers was a small percentage of the total and also does not seem to be caused because of faulty measurements/recording**.**

C. Summarizing Outcomes from Data Cleaning Steps

First, dropped columns (CaseOrder, Zip, Lat, Lng , Population, and Interaction) as upon initial inspection, didn't seem to have any relevance to the data. The outcome of this data cleaning step was so that we would reduce the amount of data we are working with, thus making it easier to handle.

Second, addressed any inconsistencies and duplications. There didn't seem to be any but the outcome of this step was to be sure that we have the most accurate data and no errors that would affect our analysis later.

Third, the imputation of the missing data for the significant quantitative variables (Age, Income , Tenure , MonthlyCharge , and Bandwidth_GB_Year) via using the median was done so that we would not have to drop the missing rows all together (decided as the percentage of missing was less then the 25% threshold for each individual variable). This outcome made us keep the data, which is valuable information for us and instead, switch it to keep the accuracy of the data intact

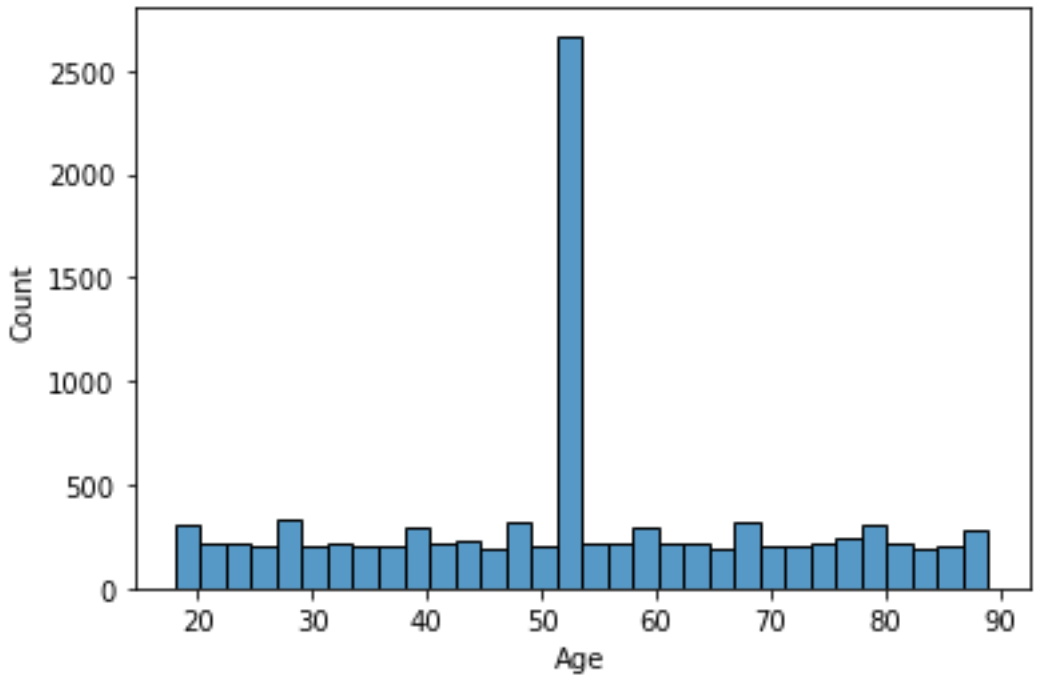
Finally, we solved for the outliers for the significant variables (Income and MonthlyCharge) but decided to keep all of them in the data. This was since the number of outliers outside the range of 3 standard deviations from the mean was small, also factoring in the case that none of them seemed to be imputed incorrectly or sourcing from an error in recording. The outcome of this again is to keep the accuracy of the data intact and not removing any data unnecessarily **Shengping Yang PHD, Gilbert Berdine MD (2016).**

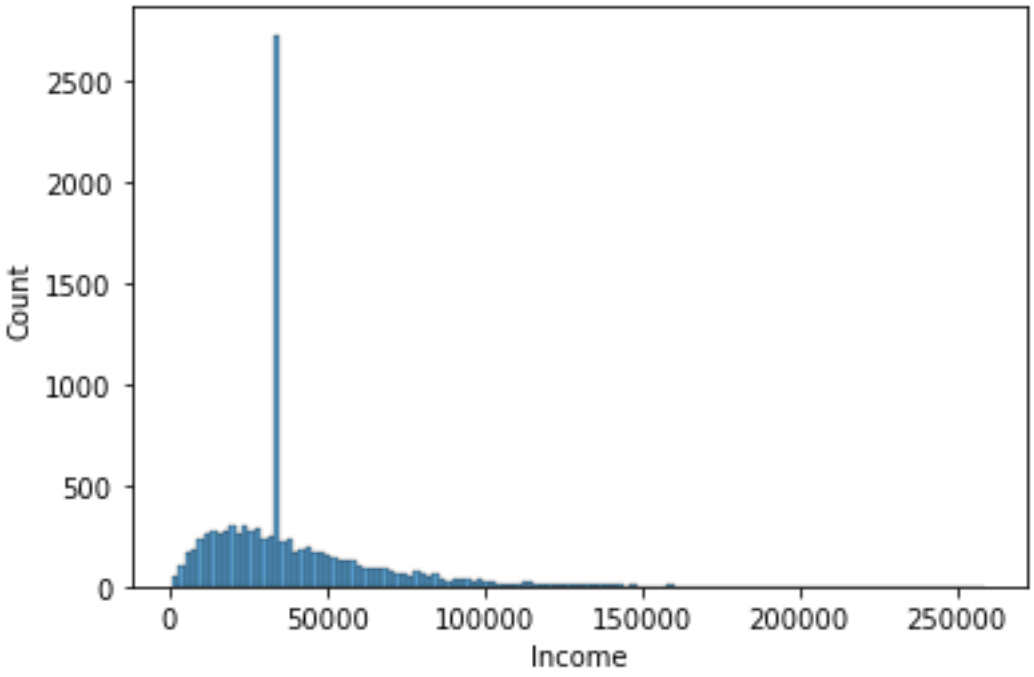
D. Visuals and Mitigation code for Anomalies

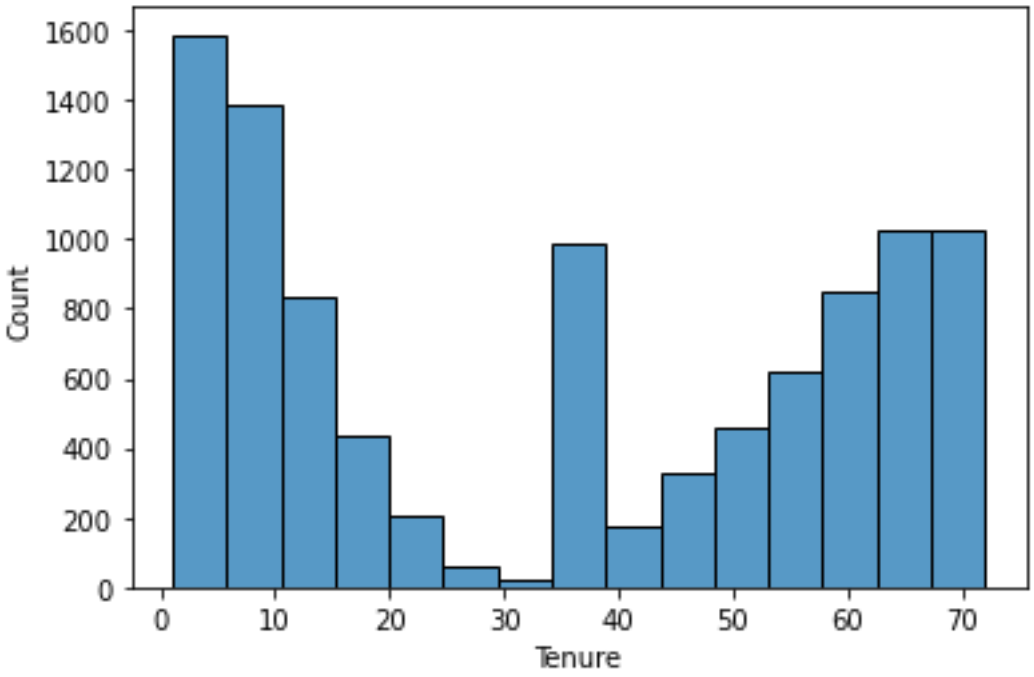
```
#imputing the missing numeric data with the Median values and imputing the missing Categorical data with the most common class
#to make all missing values zero
df_New['Children'] = df['Children'].fillna(df['Children'].median())
df_New['Age'] = df['Age'].fillna(df['Age'].median())
df_New['Income'] = df['Income'].fillna(df['Income'].median())
df_New['Tenure'] = df['Tenure'].fillna(df['Tenure'].median())
df_New['Bandwidth_GB_Year'] = df['Bandwidth_GB_Year'].fillna(df['Bandwidth_GB_Year'].median())

df_New['Techie'] = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
df_New['Phone'] = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
df_New['TechSupport'] = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

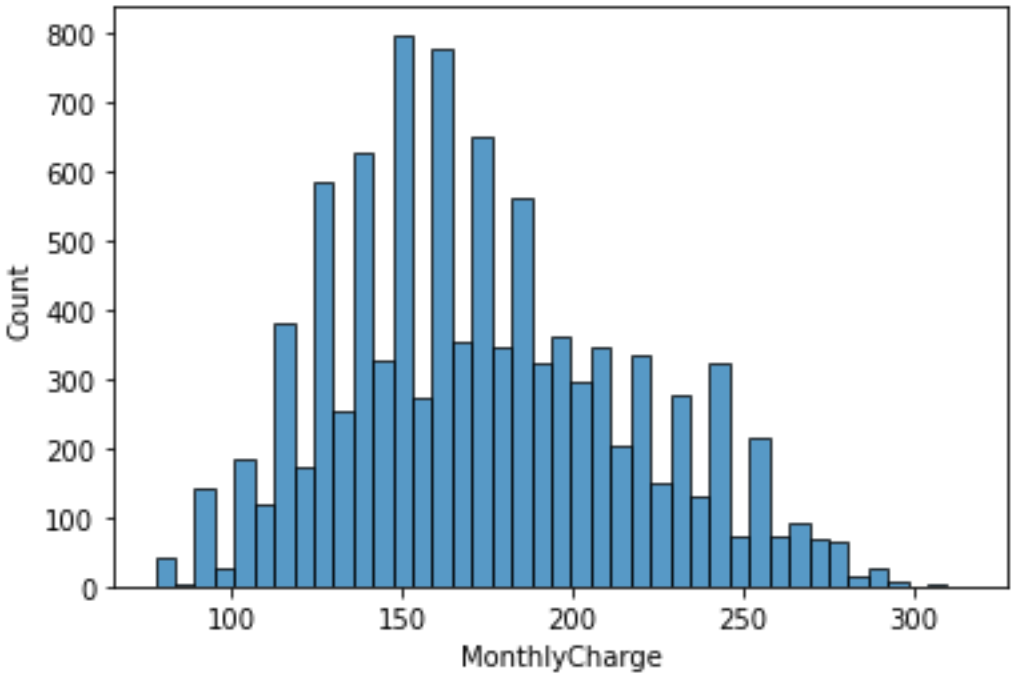
```
data_missing_after = df_New.isnull().sum()
data_missing_after
```

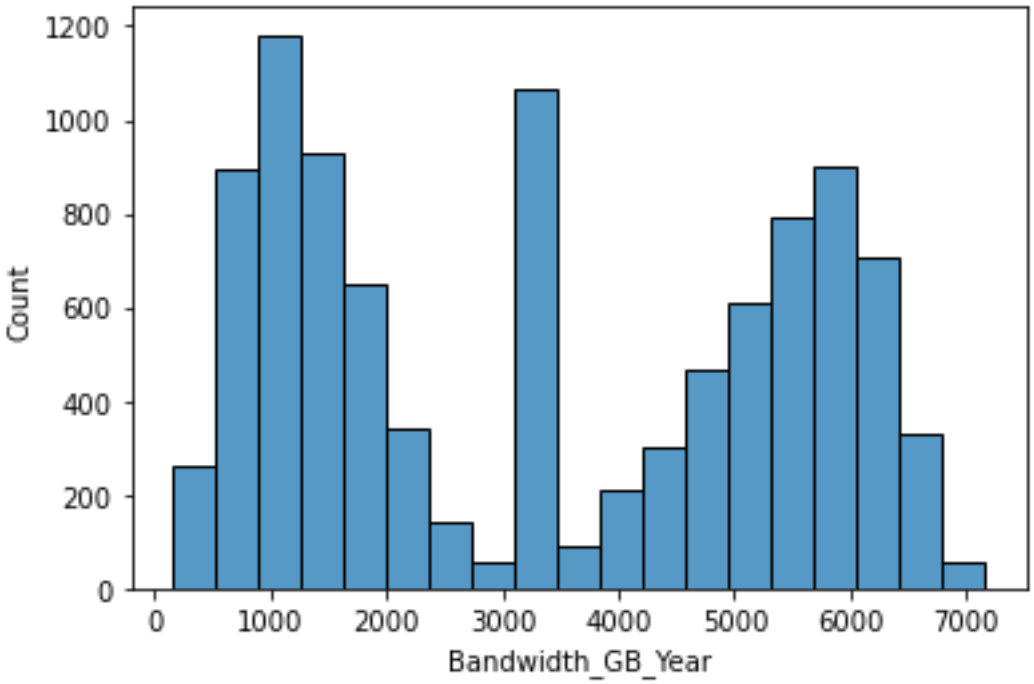
```
#Now we will utilize Histograms to take a look at our Significant variables above to see how data looks like now that the
#data has been cleaned out of missing variables
sns.histplot(data=df_New, x = 'Age')
sns.histplot(data=df_New, x = 'Income')
sns.histplot(data=df_New, x = 'Tenure')
sns.histplot(data=df_New, x = 'MonthlyCharge')
sns.histplot(data=df_New, x = 'Bandwidth_GB_Year')
```




2500
2000
1500
1000
500
0
Count
0
50000
100000
150000
200000
250000
Income

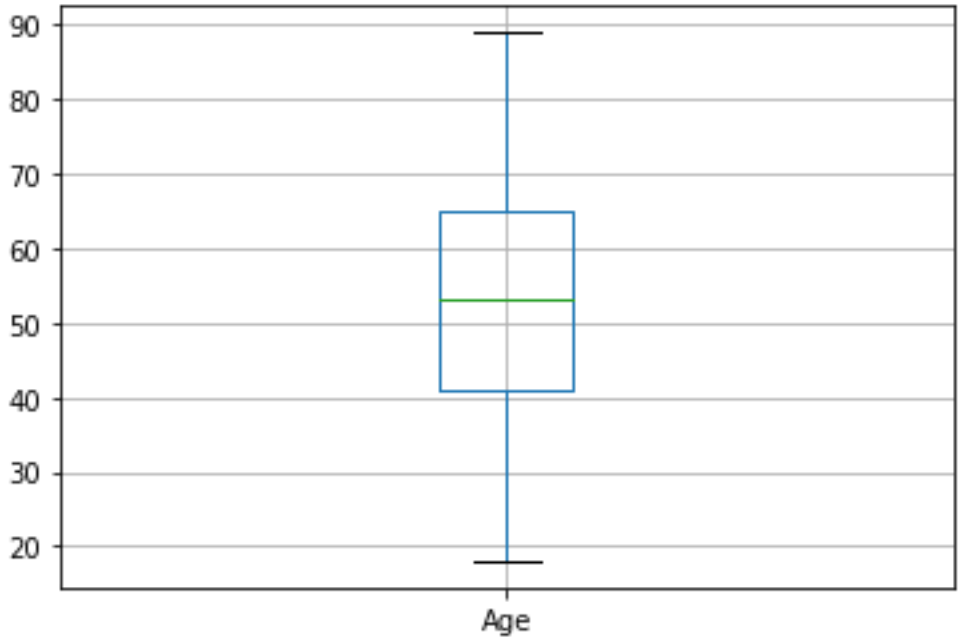

1600
1400
1200
1000
800
600
400
200
0
Count
0
10
20
30
40
50
60
70
Tenure

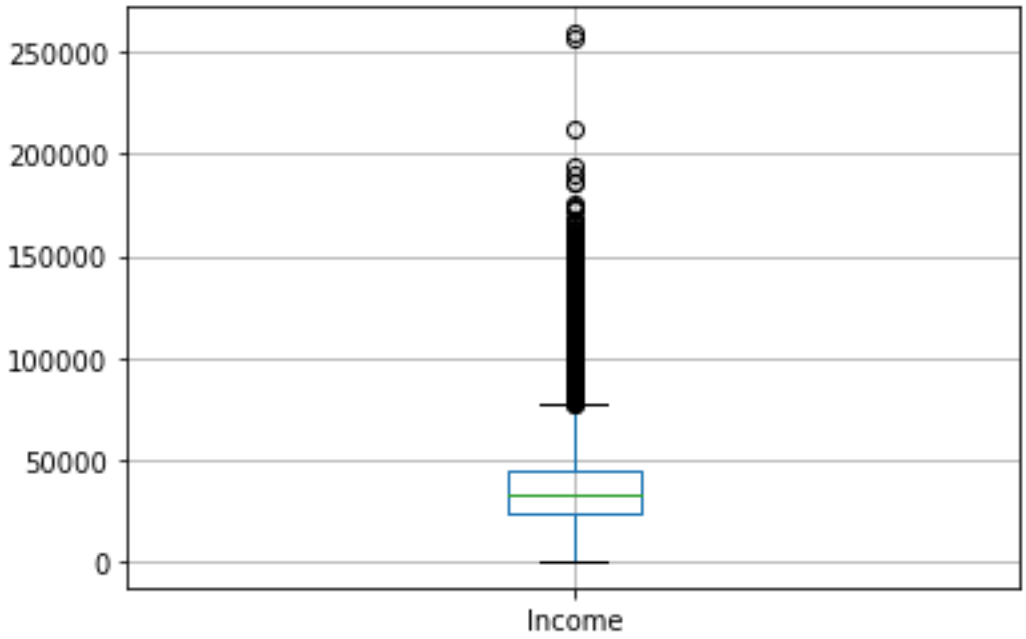

Count
MonthlyCharge
0
100
200
300
400
500
600
700
800
100
150
200
250
300

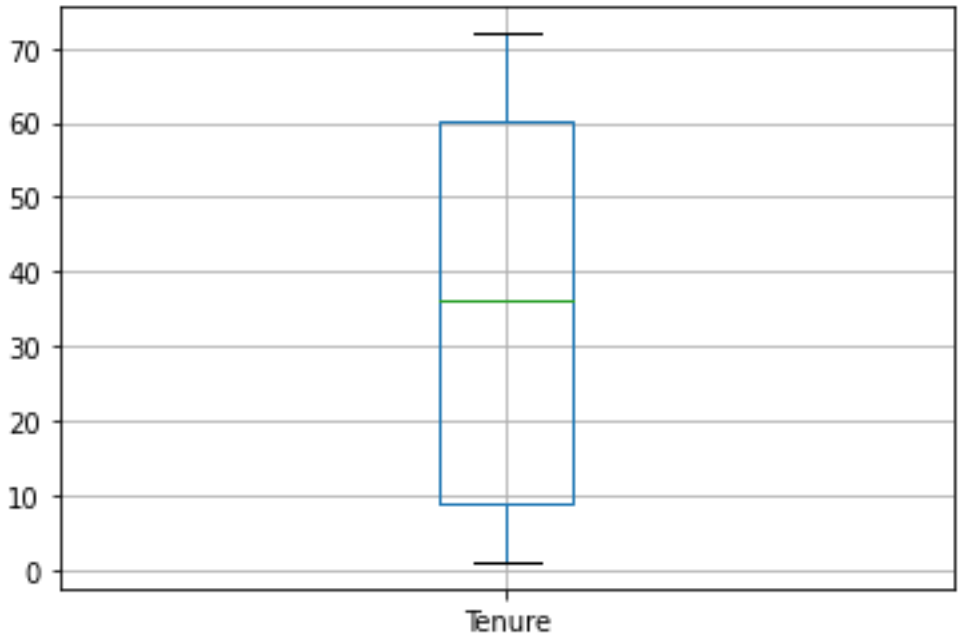

Count
Bandwidth_GB_Year
0
200
400
600
800
1000
1200
0
1000
2000
3000
4000
5000
6000
7000

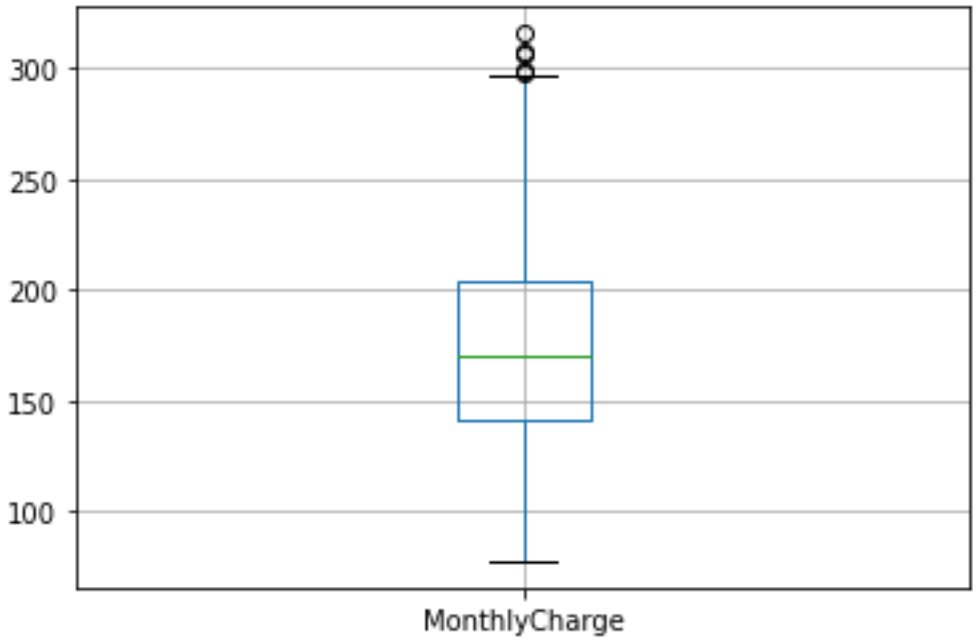
```
#For Age, it is concluded that there are no outliers present in this
variable, where outliers in this case is defined as values
#greater than 3 standard deviations away from the mean. Lets take a look at
this visually using boxplot
```

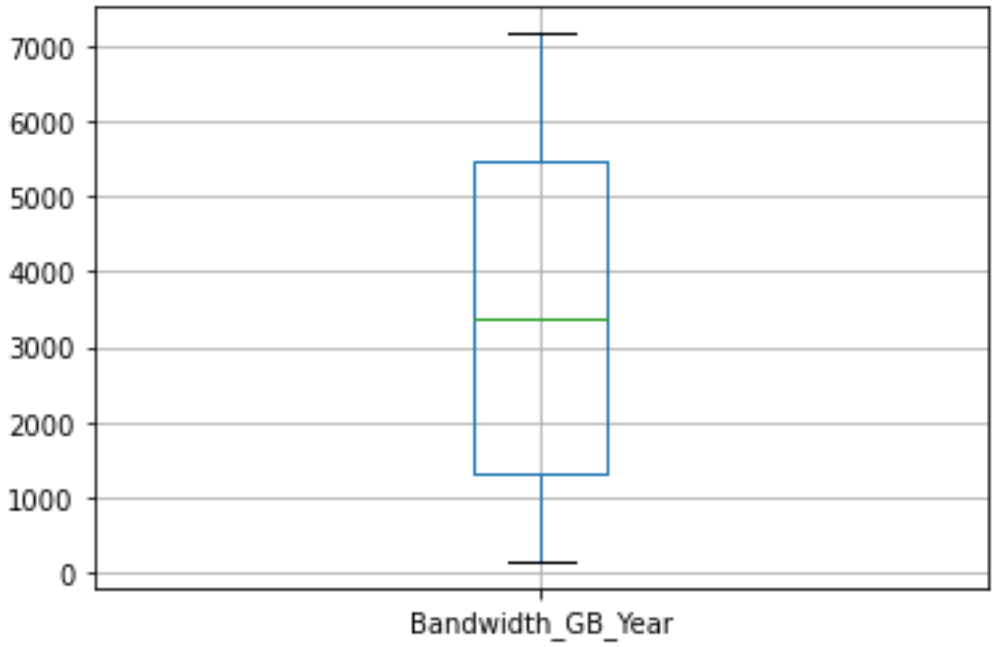
```
df_New.boxplot(['Age'])

#For Income, it is concluded that there are 193 outliers present in this
variable, where outliers in this case is defined as values
#greater than 3 standard deviations away from the mean. Lets take a look at
this visually using boxplot
df_New.boxplot(['Income'])

#For Tenure, it is concluded that there are no outliers present in this
variable, where outliers in this case is defined as values
#greater than 3 standard deviations away from the mean. Lets take a look at
this visually using boxplot
df_New.boxplot(['Tenure'])

#For MonthlyCharge, it is concluded that there are 3 outliers present in this
variable, where outliers in this case is defined as values
#greater than 3 standard deviations away from the mean. Lets take a look at
this visually using boxplot
df_New.boxplot(['MonthlyCharge'])

#For Bandwidth_GB_Year, it is concluded that there are no outliers present in
this variable, where outliers in this case is defined as values
#greater than 3 standard deviations away from the mean. Lets take a look at
this visually using boxplot
df_New.boxplot(['Bandwidth_GB_Year'])
```




250000
200000
150000
100000
50000
0
Income


70
60
50
40
30
20
10
0
Tenure






```
# Filtering out the outliers beyone and below the 3 standard deviations from
the mean for variable AGE
df2_Age_no_outlier = df_New[(df_New.Age < upper_limit_Age) & (df_New.Age >
lower_limit_Age)]
df2_Age_no_outlier.shape
# letting us see how many outliers were removed from this filtering process
for variable AGE If we decide to filter out
df_New.shape[0] - df2_Age_no_outlier.shape[0]
```

```
# Filtering out the outliers beyone and below the 3 standard deviations from the mean for variable Income
df2_Income_no_outlier = df_New[(df_New.Income < upper_limit_Income) & (df_New.Income > lower_limit_Income)]
df2_Income_no_outlier.shape
# letting us see how many outliers were removed from this filtering process for variable Income If we decide to filter out
df_New.shape[0] - df2_Income_no_outlier.shape[0]
```

```
# Filtering out the outliers beyone and below the 3 standard deviations from the mean for variable Tenure
df2_Tenure_no_outlier = df_New[(df_New.Tenure < upper_limit_Tenure) & (df_New.Tenure > lower_limit_Tenure)]
df2_Tenure_no_outlier.shape
# letting us see how many outliers were removed from this filtering process for variable Tenure If we decide to filter out
df_New.shape[0] - df2_Tenure_no_outlier.shape[0]
```

```
# Filtering out the outliers beyone and below the 3 standard deviations from the mean for variable MonthlyCharge
df2_MonthlyCharge_no_outlier = df_New[(df_New.MonthlyCharge < upper_limit_MonthlyCharge) & (df_New.MonthlyCharge > lower_limit_MonthlyCharge)]
df2_MonthlyCharge_no_outlier.shape
# letting us see how many outliers were removed from this filtering process for variable Tenure If we decide to filter out
df_New.shape[0] - df2_MonthlyCharge_no_outlier.shape[0]
```

```
# Filtering out the outliers beyone and below the 3 standard deviations from the mean for variable Bandwidth_GB_Year
df2_Bandwidth_GB_Year_no_outlier = df_New[(df_New.Bandwidth_GB_Year < upper_limit_Bandwidth_GB_Year) & (df_New.Bandwidth_GB_Year > lower_limit_Bandwidth_GB_Year)]
df2_Bandwidth_GB_Year_no_outlier.shape
# letting us see how many outliers were removed from this filtering process for variable Bandwidth_GB_Year If we decide to filter out
df_New.shape[0] - df2_Bandwidth_GB_Year_no_outlier.shape[0]
```

```
#Obtain the Cleaned Data
df_New.to_csv('Clean_Churn_Data')
```

E. Copy of Cleaned Data Set

This is attached with the assignment

F. Limitations of the Data Cleaning Process

Limitations for this data cleaning process are based around the missing values and outliers. For the missing values, I choose to impute the data with the median instead of dropping it. I would have wanted to be able to see what the reason was for the missing values and if possible if there is a database/warehouse where this actual information resides. Due to the imputation of median values from this cleaning process, the data outcome might have been affected.

For the outliers, there were several data points that were outside the range of 3 standard deviations from the mean. This means that it was located outside 99.7% of the expected data set. However I decided to keep this in since while looking at the values, it does not seem that it was incorrectly input/came from an incorrect source and might be vital to describing the data. I would like to be able to again confirm this with the company and their data warehouse to see if this was truly the case. Thus, due to leaving the outliers in the data set from the cleaning process, the outcome might have been affected.

From these two limitations, we see that It is the uncertainty from the missing values and outliers and lack of ability to verify/confirm these values with the original source inside the company's data warehouse/ employees responsible for recording the accurate data.

G. How would these limitations affect the Analysis

The affect of imputing the missing data via utilizing the median values of the individual variables can lead to the analysis having less variability since we are creating a full data set in which to analyze from. However along with the reduced variability may come along systematic bias compared to more organically obtained data as we are filling the missing data with an probably best guess of what it could have been. It is important to also note that the imputation also does not address the issue of why the missing data occurred in the first place, which could lead to more uncertainty of analysis of decision making down the line and should be thoroughly investigated.

The affect of keeping the few outliers in the variables Income and MonthlyCharge on the data can lead to more insights in the analysis phase since we can consider extreme valid observations that occur through variability of the data. Of course, in an ideal world we could confirm with the data warehouse collecting these values if the outliers caused were valid or from unknown errors. This cause we assume validity.

H. Utilizing the PCA to identify the Significant Variables

```
#Prepare for the PCA by removing all categorical data and only leaving
numerical data
churn_data = pd.read_csv('Clean_Churn_Data')

churn_data =
churn_data[['Children','Age','Income','Outage_sec_perweek','Email','Contacts
','Yearly_equip_failure','Tenure','MonthlyCharge','Bandwidth_GB_Year','Timel
y_Response','Timely_Fixes', 'Timely_Replacements',

'Reliability','Options','Respectful_Response','Couteous_Exchange',
                          'Evidence_Of_Active_Listening']]

churn_data.head()
```

```
#Time to go through the PCA process, first by importing the necessary library
from sklearn.decomposition import PCA
```

```
#Normalizing the data
Normalized_Churn = (churn_data - churn_data.mean())/churn_data.std()
```

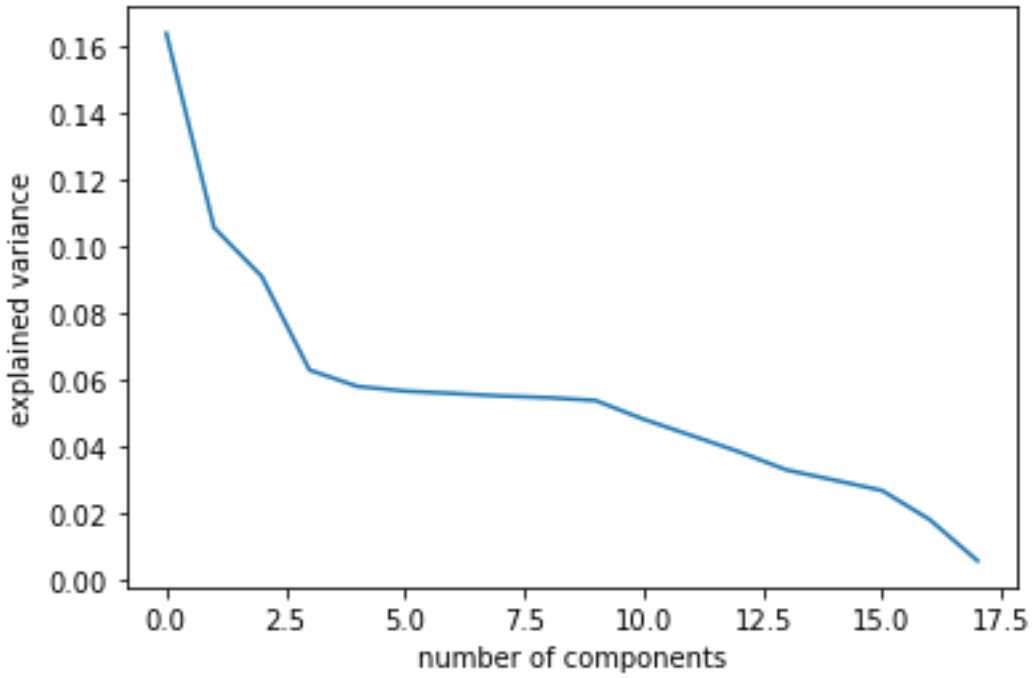
```
#we want to ensure that all componenets are extracted from the analysis so we
use the following function
pca = PCA(n_components = churn_data.shape[1])
```

```
#Create PCA Names
data_numeric = churn_data[
['Children','Age','Income','Outage_sec_perweek','Email','Contacts','Yearly_equ
ip_failure','Tenure','MonthlyCharge','Bandwidth_GB_Year','Timely_Response','Ti
mely_Fixes', 'Timely_Replacements',

'Reliability','Options','Respectful_Response','Couteous_Exchange',
                          'Evidence_Of_Active_Listening']]
pcs_names = []
for i, col in enumerate(data_numeric.columns):
    pcs_names.append('PC' + str(i+1))
print(pcs_names)
```

```
#Utilize the PCA application to create a new data set and convert the 18
variables into 18 components
pca.fit(Normalized_Churn)
Churn_pca = pd.DataFrame(pca.transform(Normalized_Churn),
columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','
PC12','PC13','PC14','PC15','PC16','PC17','PC18'])
```

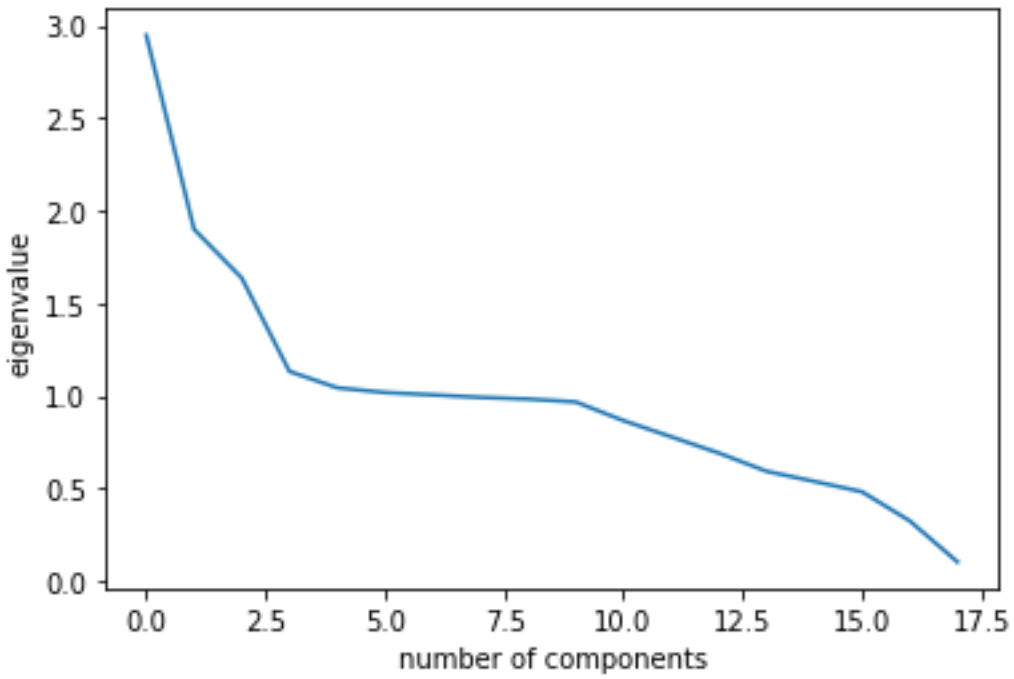
```
#import graphing libraries before continuing on to make a scree plot
import matplotlib.pyplot as plt

import seaborn as sns
```

```
#create the scree plot
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('number of components')
plt.ylabel('explained variance')
plt.show()
```




```
#Now we will view the scree plot with eigen values to get a more indepth look
cov_matrix = np.dot(Normalized_Churn.T, Normalized_Churn) /
churn_data.shape[0]
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for
eigenvector in pca.components_]
```

```
plt.plot(eigenvalues)
plt.xlabel('number of components')
plt.ylabel('eigenvalue')
plt.show()
```




```
#we now select the fewest components to make a reduction decision for the data
to only include data that captures 90% of original
#data set variance
for pc, var in zip(pcs_names,np.cumsum(pca.explained_variance_ratio_)):
   print(pc,var)
```

```
#From the above, it can be concluded that 11 components can accurately explain
variability of about 92% of the data
#We now create a rotation to find which variable has the highest load
magnitude in the PCA Algorithm
rotation = pd.DataFrame(pca.components_.T, columns = pcs_names, index =
data_numeric.columns)
print(rotation)
```

```
# Obtian the Output Loadings
loadings = pd.DataFrame(pca.components_.T,
columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','
PC12','PC13','PC14','PC15','PC16','PC17','PC18'],
index=churn_data.columns)
loadings
```

```
data_reduced = Churn_pca.iloc[:,0:3]
print(data_reduced)
```

I. List of Principle Components in the Data Set

The principle components in the data set that were given the most magnitude via the rotation analysis and were deemed most significant through our PCA procedure are as follows: Timely Response, Respectful Response, Timely Fixes, Timely Replacements, and Tenure.

J. How Principle Components were Identified

The principal components of the data set were obtained and identified through the use of the PCA analysis.

1) First, we prepped the data to only have numerical variables present. This required us to reduce the original cleaned data frame to a smaller one.
2) We then normalized the data to avoid the PCA loading on the variables that happen to have a larger variance
3) We then created the 18 total numerical variables we had into 18 components
4) Utilized the Scree plot and Eigen value plot to visually see roughly where the elbow bends at the optimal amount of components that explains a significant percentage of the variance. In this case the elbow bent at roughly 3 components
5) After concluding that 11 components explained 92% of the data, we then did a variable loading analysis to see which specific variables were of importance of the already reduced 3 component
6) This gave us the results of the most significant variables and components in the data set to focus on

K. Organizational Benefits from PCA Analysis

This organization can benefit greatly from the results of the PCA since they will be able to know which indicators and metrics to pay specific attention to. We see that the majority of the metrics in which Churn is affected by are with customer service rather than actual plans or product capabilities. This does make sense since customer service (Timely Response, Respectful Response, Timely Fixes, Timely Replacements) are what infact separates most companies offering similar products. Thus addressing how the company handles customer issues would significantly reduce their customer churn. Ofcourse with dealing with Tenure and Bandwidth_GB_Year, this is also important to focus on according to the analysis since the more Tenure increases, the more customers have a built up loyalty with the company and less likely to churn.

We can say that just by looking at these variables, it makes sense but now we have concrete data indicating that these are the strongest factors when considering churn rate, thus if the company can play their cards right and improve some of these aspects, they will definitely see a reduction in their rate of customer churn.

Part IV. Supporting Documents

A. Panopto Video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=67b55617-72c9-45b5-a06b-ad460001ac59

B. Third Party Code References

Lianne & Justin (2020). Data Cleaning in Python: The Ultimate Guide
https://towardsdatascience.com/data-cleaning-in-python-the-ultimate-guide-2020-c63b88bf0a0d

StackOverFlow (2018). Find out Percentage of Missing Values in each Column in the given Data Set
https://stackoverflow.com/questions/51070985/find-out-the-percentage-of-missing-values-in-each-column-in-the-given-dataset/51071037

StackOverFlow (2015). Imputation of Missing Values for Categories in Panda
https://stackoverflow.com/questions/32617811/imputation-of-missing-values-for-categories-in-pandas

StackOverFlow(2017). Factor Loadings using sklearn
https://stackoverflow.com/questions/21217710/factor-loadings-using-sklearn

Seaborn (2021). Seaborn.Histplot https://seaborn.pydata.org/generated/seaborn.histplot.html

Seaborn(2021). Seaborn.Boxplot https://seaborn.pydata.org/generated/seaborn.boxplot.html

C. Acknowledged Sources References

DataCamp (2020). Choosing Python or R for Data Analysis? An Infographic. DataCamp. https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis

Yildirim, S. (2021). The Most Helpful Python Data Cleaning Modules. LearnPython. https://learnpython.com/blog/python-libraries-data-cleaning/

Seaborn (2021). Seaborn: Statistical Data Visualization. Seaborn. https://seaborn.pydata.org/

Anthony B. Ryder, Anna V. Wilkinson, Michelle K. McHugh, Katherine Saunders, Sumesh Kachroo, Anthony D'Amelio, Jr., Melissa Bondy, and Carol J. Etzel (2011). The Advantage of Imputation of Missing Income Data to Evaluate the Association Between Income and Self-Reported Health Status (SRH) in a Mexican American Cohort Study. NCBI. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3205225/

Shengping Yang PHD, Gilbert Berdine MD (2016). Outliers. Pulmonary Chronicles. https://pulmonarychronicles.com/index.php/pulmonarychronicles/article/view/252/635