D212 Performance Assessment Task 2

This is my task 2 for D212, focused on PCA.012047746

A1:PROPOSAL OF QUESTION

The research question I have for this project is whether we can use PCA analysis to reduce down the numerical continuous features we have about customer to the most important ones. This is an important business question to answer because we can understand what data we have about customers that is redundent. It allows us to reduce dimensionality which helps us create new types of analysis. Once this is done, we can answer business questions to a more accurate degree with less redundency in our inputted variables. For instance, we can create more accurate segmentation from these principal components which will allow us to make better decisions related to groups of customers.

A2:DEFINED GOAL

The defined goal for this analysis is to use PCA on the continuous numerical variables of the dataset to reduce down the features of the dataset, get the component variance of each of the significant principle components, and get the total variance which will help explain what amount the significant principle components can capture of the original dataset.

B1:EXPLANATION OF PCA

PCA analyzes the selected dataset by reducing it to principal components. A PCA plot is created from this which converts the correlations among all of the columns into a 2d graph. When they are highly correlated, columns will cluster togther. The axes are also ranked in order of importance. The first principal component axis will be more important than the differences along the second principal component axis.

Important to PCA is the normalization of data. I will begin by normalizing the data. Then I will create a loading matrix from this normalized data. This loading matrix shows us the weights for each variable in association with each priniciple component. It allows us to understand the significance of a specific variable on a certain principle component. Then we use the elbow method or kaiser criterion to create a

scree plot that explains which principle components are significent. At this point we have reduced dimensionality. Then we can calculate the component's variance and the total variance. That total variance will explain the amount that the significant principle components can explain the total variability of the original dataset.

B2:PCA ASSUMPTION

Some assumptions of PCA are:

- PCA assumes that there is a correlation between features. If there is no correlation between features, the model will be unable to create principal components
- PCA assumes that the data has been normalized. If the data is not normalized, the scale of the data will affect how PCA is created
- PCA assumes a linear relationship between features. It can not determine a relationship like exponential for instance as well

C1:CONTINUOUS DATA SET VARIABLES

We will start by importing the libraries and data that we need:

- Pandas: Pandas is important as it add in dataframes which allows us to import csv's, modify their data, and input it into models
- Numpy: Numpy is used in this analysis in order to read data from a column in a dataframe. Numpy is used to work with data in an array format
- Scipy: Scipy is used in this analysis for finding z-score.
- Matplotlib: Matplotlib is used to plot graphs
- Seaborn: Seaborn is used to visualize data similar to matplotlib
- Sklearn: User for normalization and for performing the principal component analysis

In [128…

```
# import libraries
import pandas as pd
import numpy as np
import csv
from sklearn.preprocessing import StandardScale
from sklearn.decomposition import PCA

import scipy.stats as stats

# import visualization tools
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [129…

```
df = pd.read_csv('churn_clean.csv')
```

In [130…

```
df.head()
```

Out[130]:

	CaseOrder	Customer_id	Interaction	
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b29988
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef96
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a11

5 rows × 50 columns

Next we will remove all categorical and discrete numerical values so we are only left with continuous columns.

In [131…
```
df_n = ['Lat','Lng','Income','Outage_sec_perwee
```

In [132…
```
df = df[['Lat','Lng','Income','Outage_sec_perwe
```

The data set variables that I will use for this are the continuous numerical variables from the original dataset. These include:

- Lat - This is a continuous variable as it is numerical
- Lng - This is a continuous variable as it is numerical
- Income - This is a continuous variable as it is numerical
- Outage_sec_perweek - This is a continuous variable as it is numerical
- Tenure - This is a continuous variable as it is numerical
- MonthlyCharge - This is a continuous variable as it is numerical
- Bandwidth_GB_Year - This is a continuous variable as it is numerical

In [133…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Lat                 10000 non-null  float6
4
 1   Lng                 10000 non-null  float6
4
 2   Income              10000 non-null  float6
4
 3   Outage_sec_perweek  10000 non-null  float6
4
 4   Tenure              10000 non-null  float6
4
 5   MonthlyCharge       10000 non-null  float6
4
 6   Bandwidth_GB_Year   10000 non-null  float6
4
dtypes: float64(7)
memory usage: 547.0 KB
```

In [134…

```
df.describe()
```

Out[134]:

	Lat	Lng	Income	Ou
count	10000.000000	10000.000000	10000.000000	
mean	38.757567	-90.782536	39806.926771	
std	5.437389	15.156142	28199.916702	
min	17.966120	-171.688150	348.670000	
25%	35.341828	-97.082812	19224.717500	
50%	39.395800	-87.918800	33170.605000	
75%	42.106908	-80.088745	53246.170000	
max	70.640660	-65.667850	258900.700000	

C2:STANDARDIZATION OF DATA SET VARIABLES

We will quickly run tests to verify that the data is clean and free of outliers before we normalize

Treat Nulls

There are no nulls in this dataset

In [135…

```
# check the data for nulls
df.isnull().sum()
```

Out[135]:
```
Lat                   0
Lng                   0
Income                0
Outage_sec_perweek    0
Tenure                0
MonthlyCharge         0
Bandwidth_GB_Year     0
dtype: int64
```

Treat Duplicates

There are no duplicates in this dataset

In [136…
```
df.duplicated().value_counts()
```

Out[136]:
```
False    10000
dtype: int64
```

Treat Outliers

We can start by checking the histograms of all of our quantiative variables. After looking through it, the distributions are as follows:

- Lat - This is a normal distribution
- Lng - This is a left skewed distribution
- Income - This is a right skewed distribution
- Outage_sec_perweek - This is a normal distribution
- Tenure - This is a bimodal distribution
- MonthlyCharge - This is a normal distribution
- Bandwidth_GB_Year - This is a bimodal distribution

This is useful information to note for later. We can also identify from our histograms if the data passes 3 standard deviations. I will use that as a cutoff for what we identify as outliers. Using this benchmark, the following variables contain outliers:

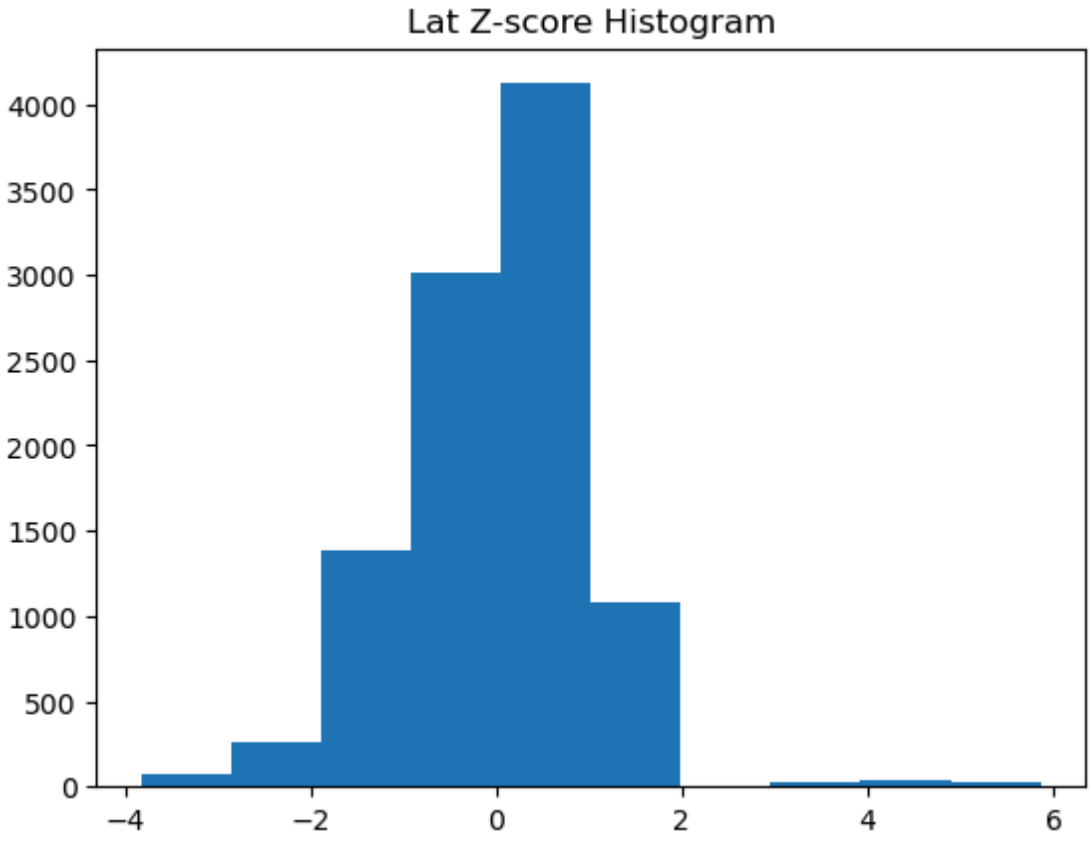
- Lat
- Lng
- Income

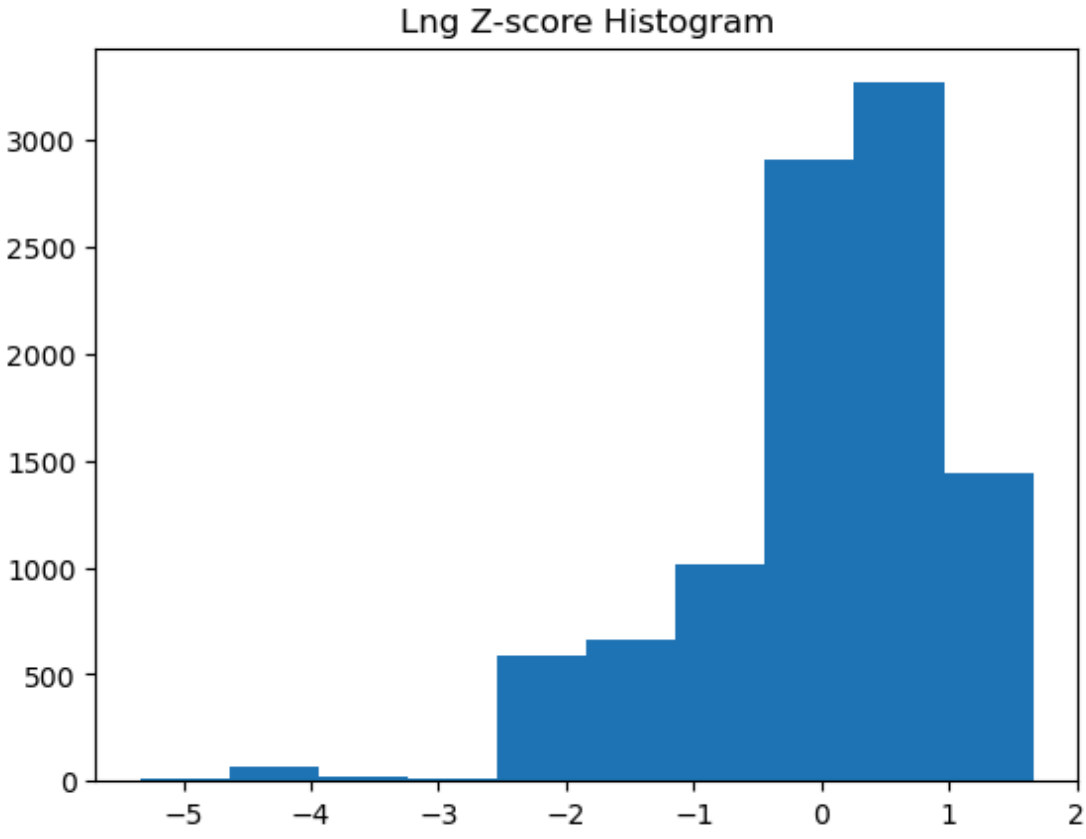
Now that I know what variables are the ones that need to be solved, I can run a for loop to drop the outliers which are values equivalent to a z-score greater or less than 2 and -2 respectively. We also need to know the distribution to understand what we need to impute these variables with.

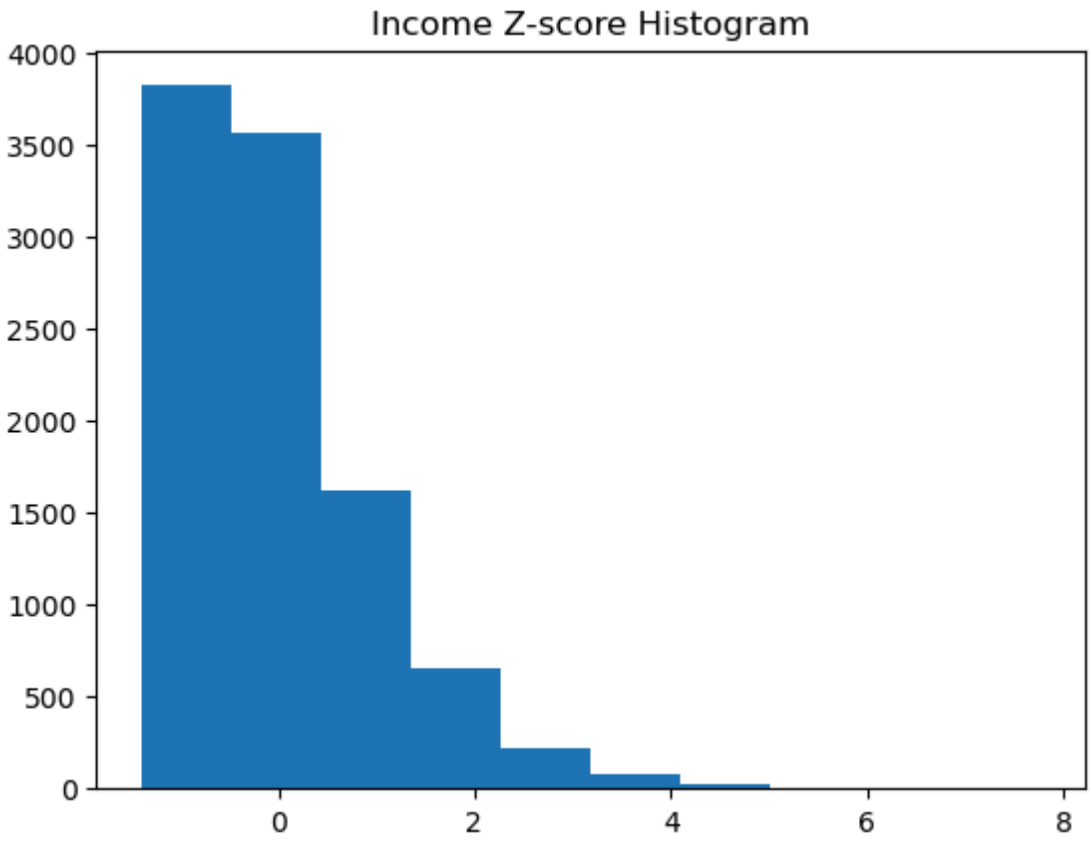
- Normal - replace with mean
- Right or left skewed - replace with median

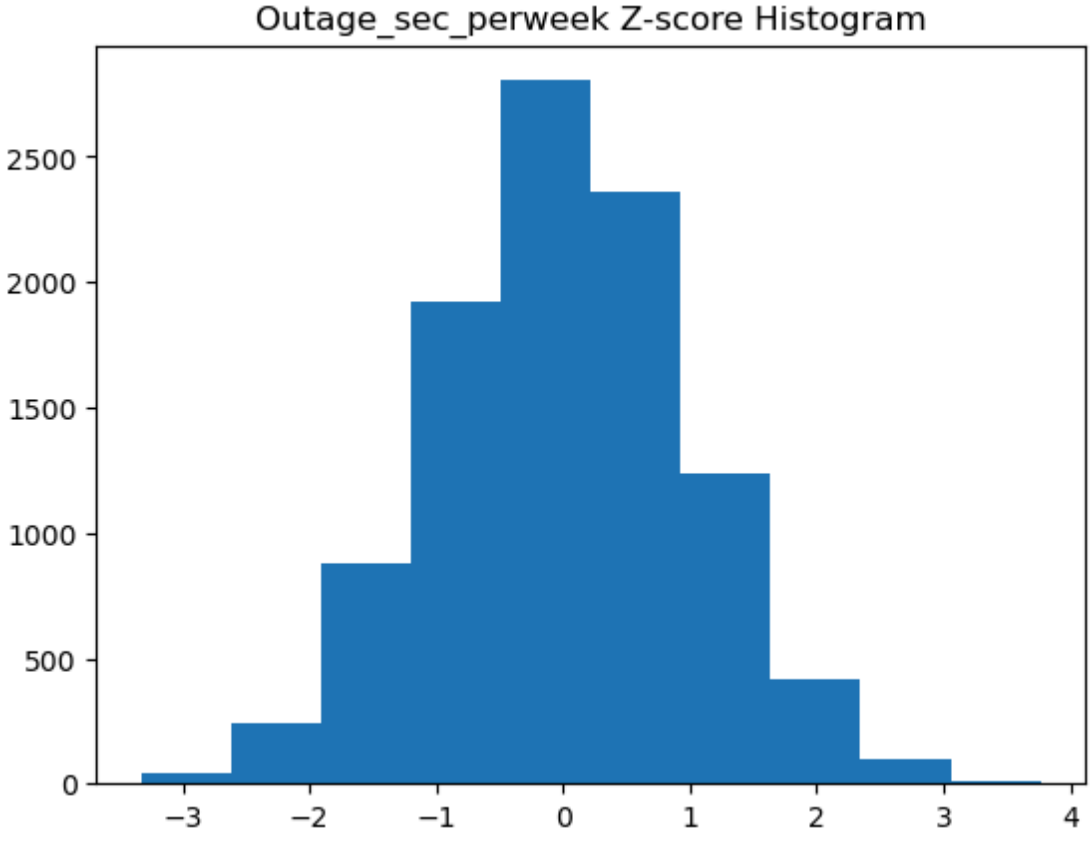
In [137…

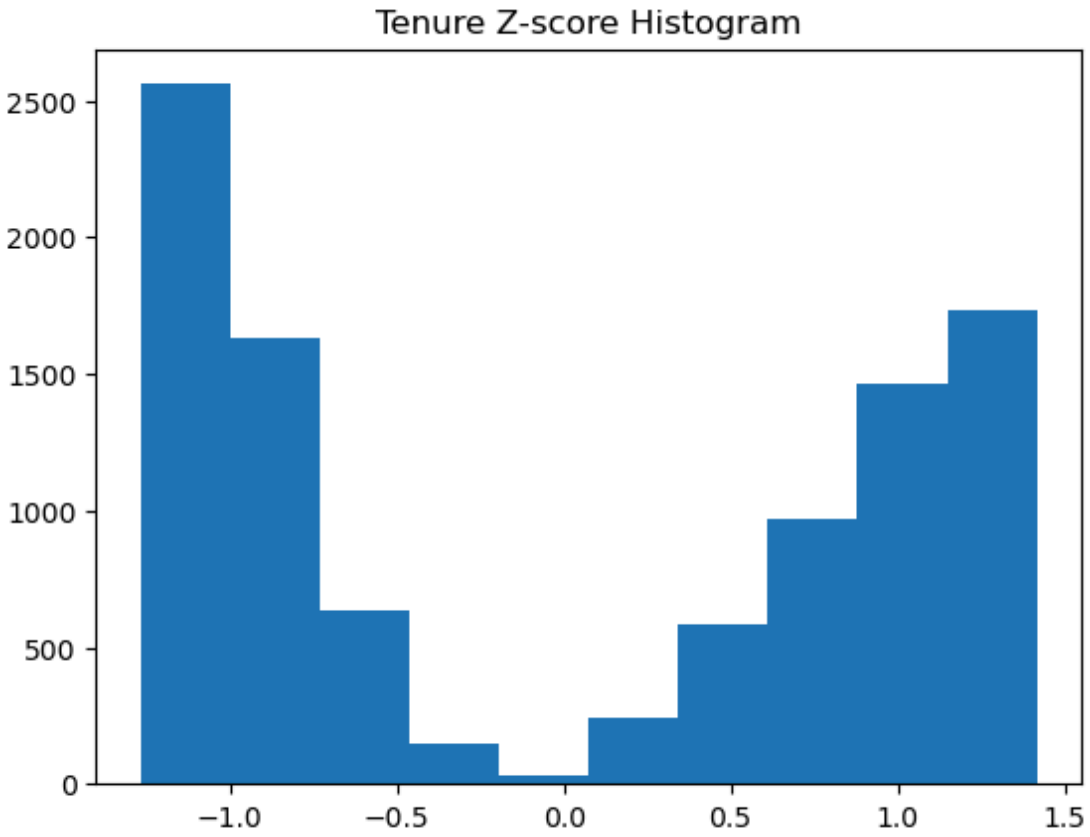
```
for column in df_n:
    df['zscore'] = stats.zscore(df[column])
    plt.hist(df['zscore'])
    plt.title(column + ' Z-score Histogram')
    plt.show()
```

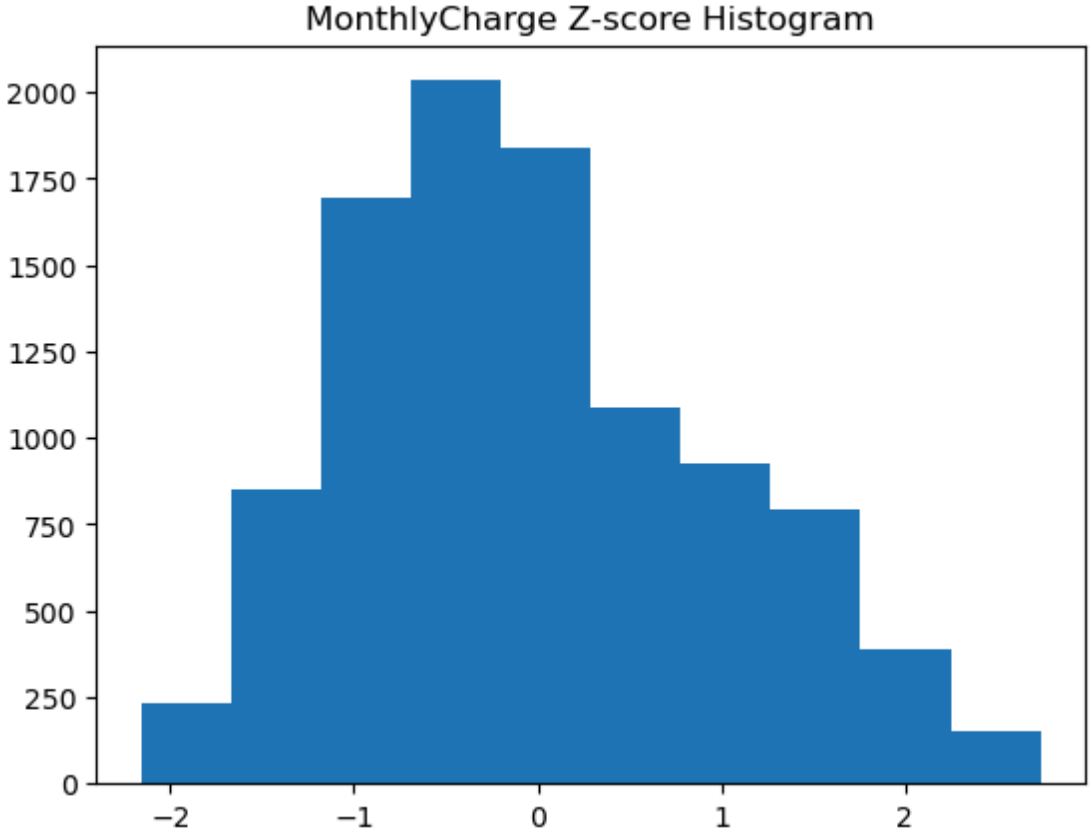

Lat Z-score Histogram
4000
3500
3000
2500
2000
1500
1000
500
0
−4
−2
0
2
4
6

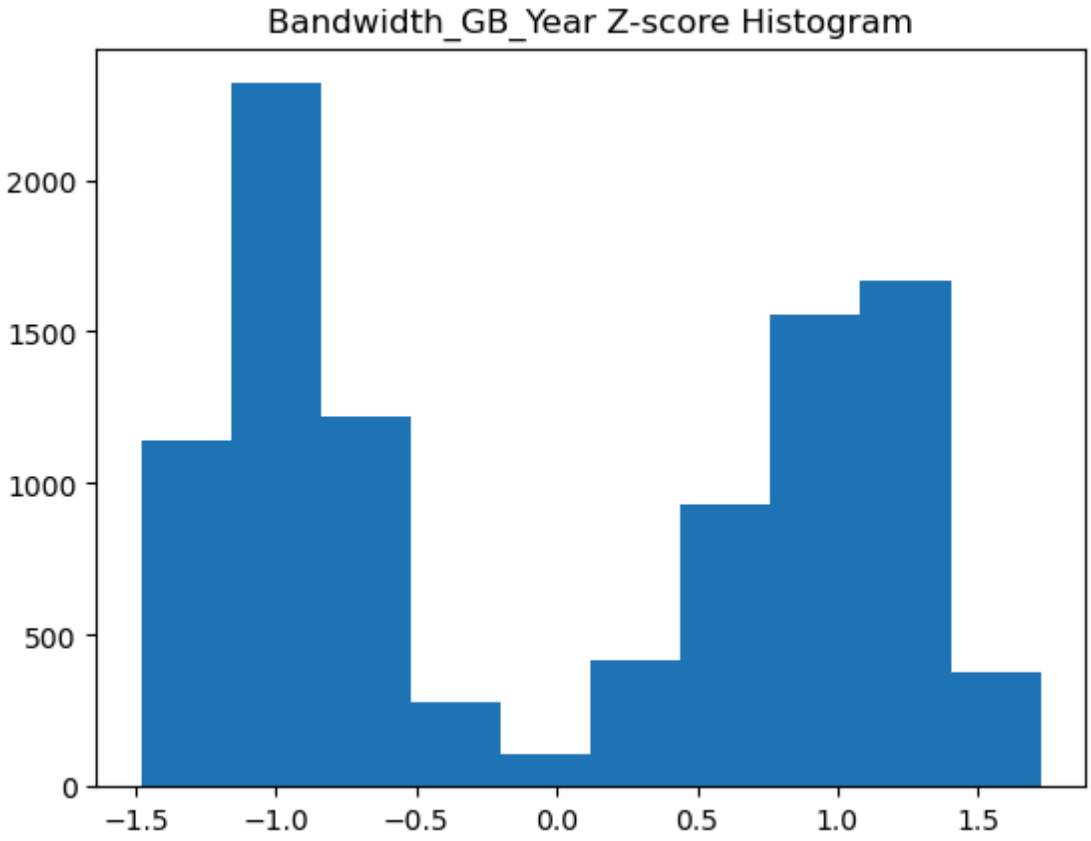

Lng Z-score Histogram
3000
2500
2000
1500
1000
500
0
−5
−4
−3
−2
−1
0
1
2


Income Z-score Histogram
4000
3500
3000
2500
2000
1500
1000
500
0
0
2
4
6
8


Outage_sec_perweek Z-score Histogram
2500
2000
1500
1000
500
0
−3
−2
−1
0
1
2
3
4


Tenure Z-score Histogram
2500
2000
1500
1000
500
0
−1.0
−0.5
0.0
0.5
1.0
1.5


MonthlyCharge Z-score Histogram
2000
1750
1500
1250
1000
750
500
250
0
−2
−1
0
1
2



In [138…

```
# Run a for loop for all the identified variabl
df_z_median = ['Lng','Income']
df_z_median
for column in df_z_median:
    # create nulls for outliers in population
    df['zscore'] = stats.zscore(df[column])
    df[column] = np.where(df['zscore'] > 2, np.
    df[column] = np.where(df['zscore'] < -2, np
    # use fillna function to impute outliers wi
    df[column] = df[column].fillna(df[column].m
    df = df.drop('zscore',axis=1)
```

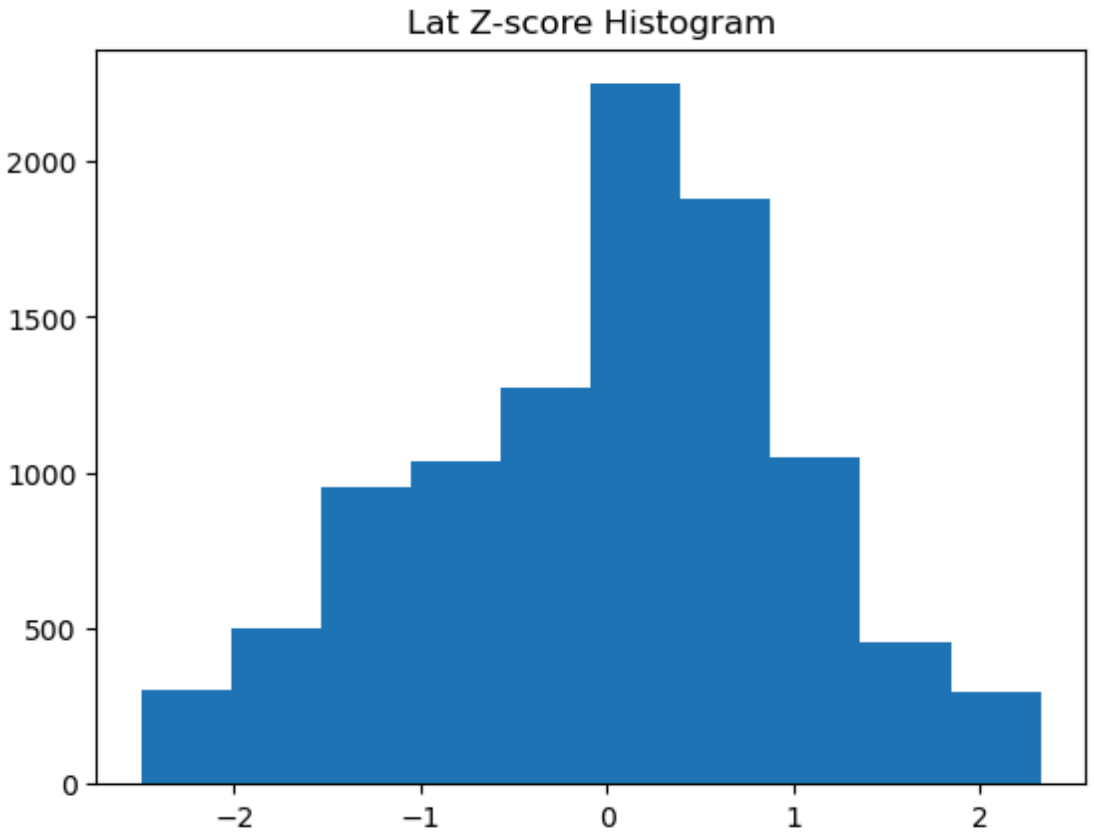
In [139…

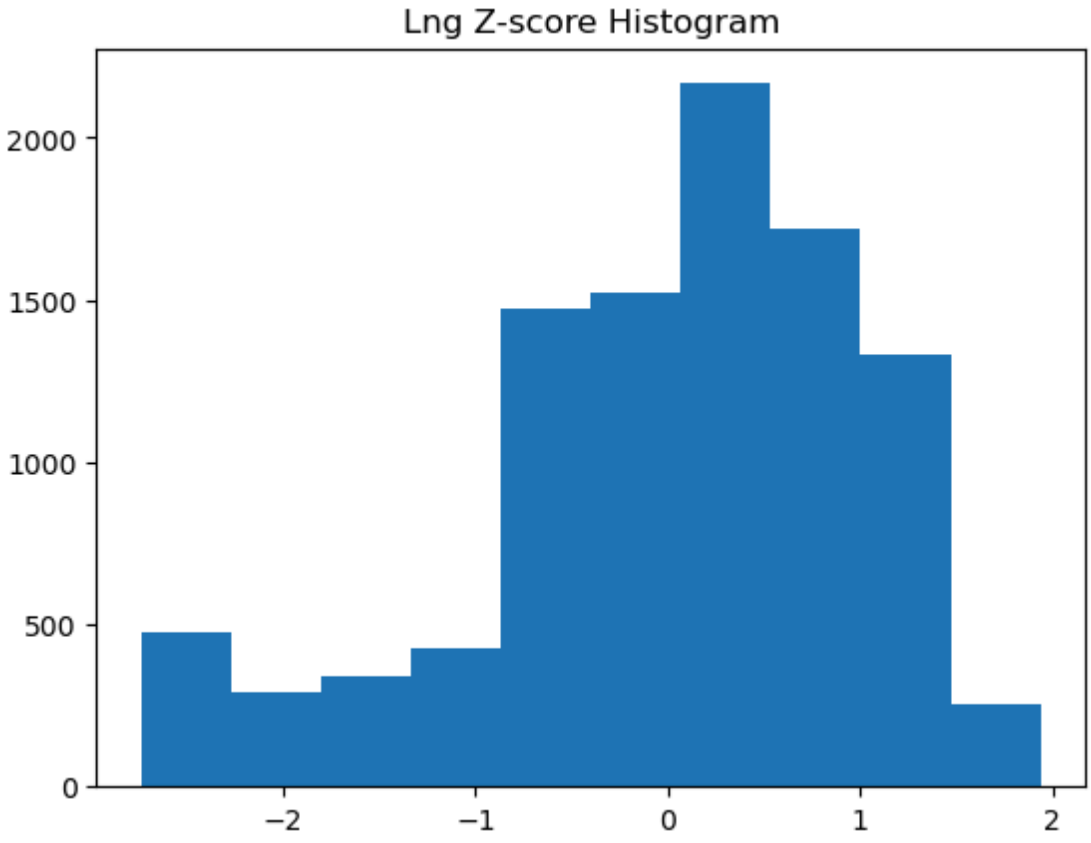
```
# Run a for loop for all the identified variabl
df_z_mean = ['Lat','Outage_sec_perweek']
df_z_mean
for column in df_z_mean:
    # create nulls for outliers in population
    df['zscore'] = stats.zscore(df[column])
    df[column] = np.where(df['zscore'] > 2, np.
    df[column] = np.where(df['zscore'] < -2, np
```

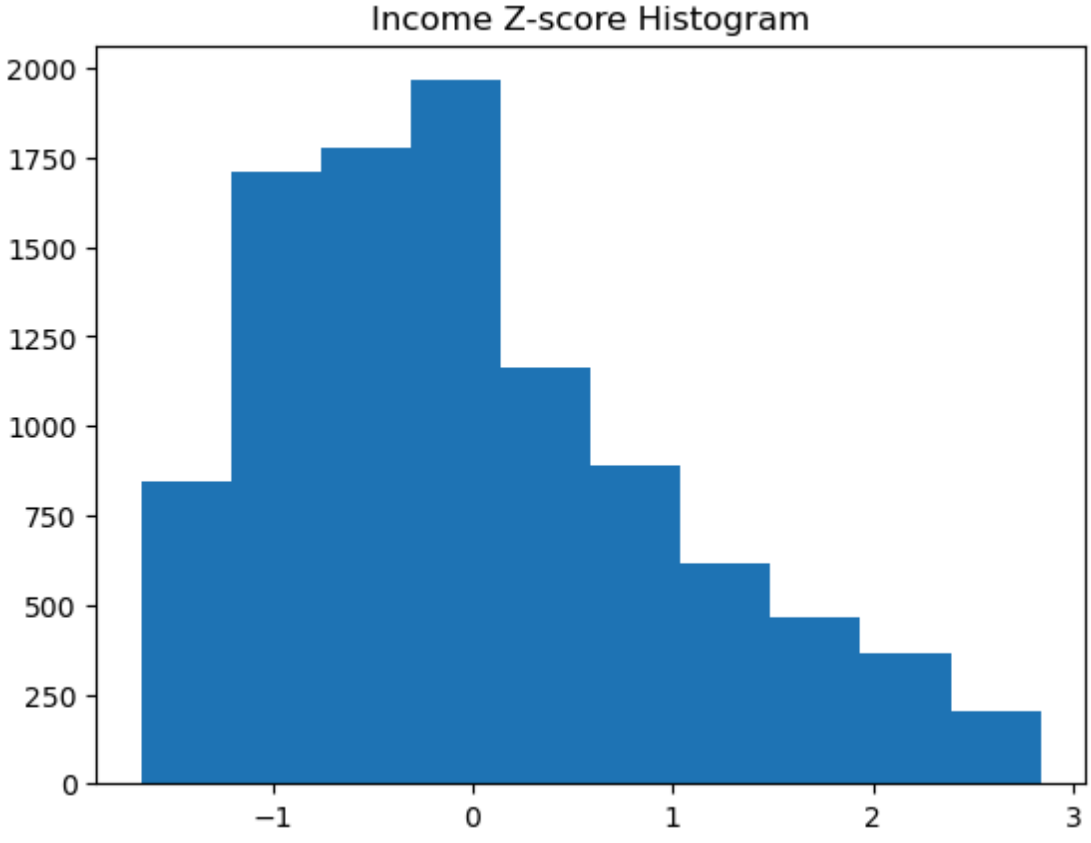
```
    # use fillna function to impute outliers wi
    df[column] = df[column].fillna(df[column].m
    df = df.drop('zscore',axis=1)
```

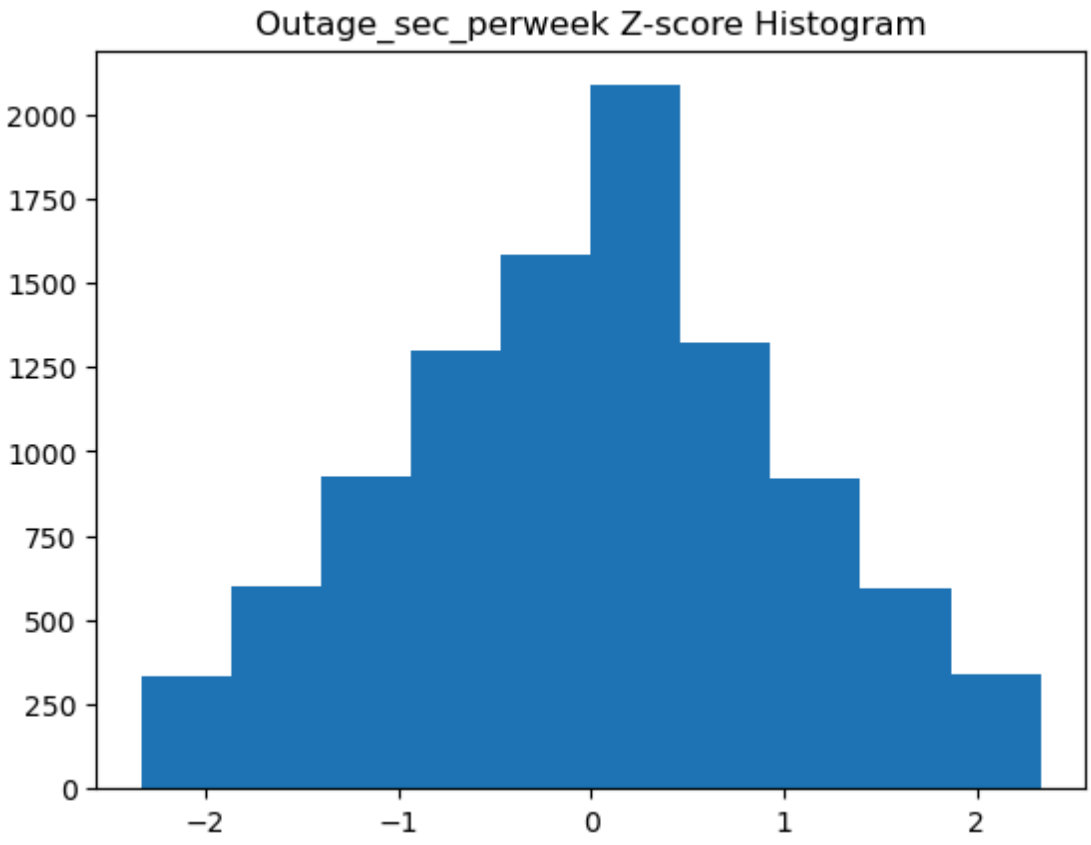
In [140…

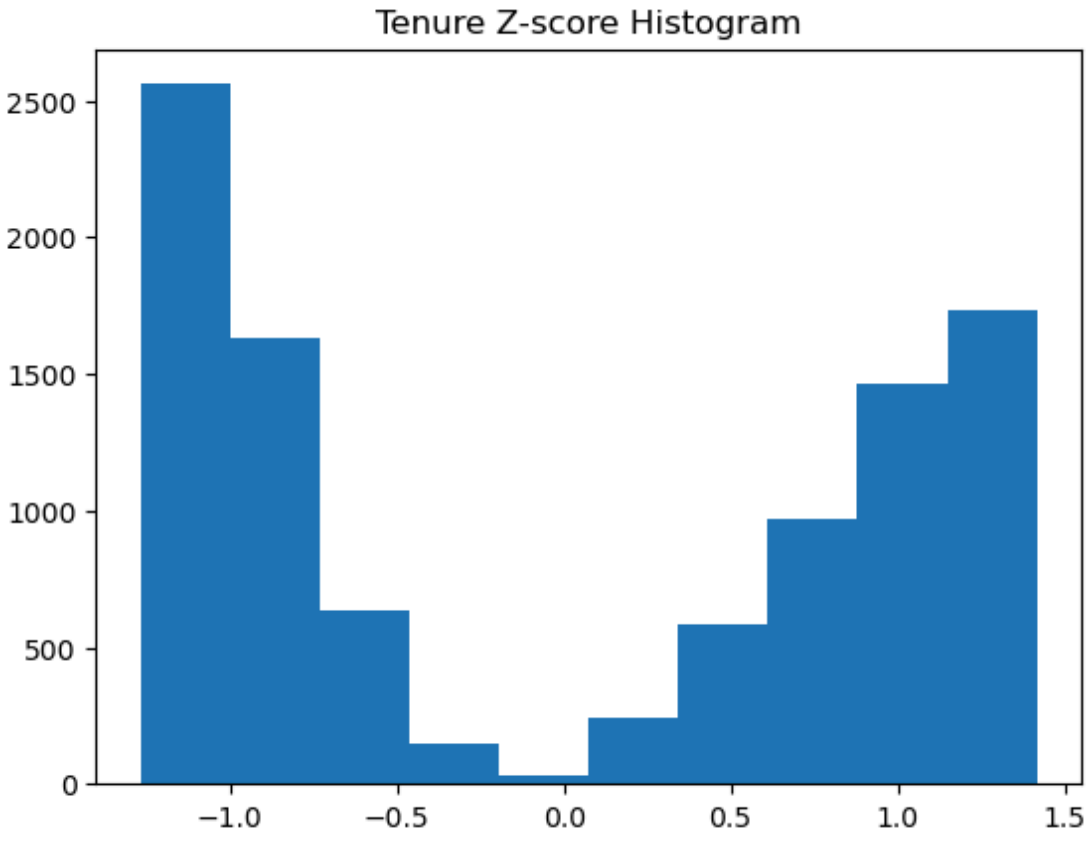
```
for column in df_n:
    df['zscore'] = stats.zscore(df[column])
    plt.hist(df['zscore'])
    plt.title(column + ' Z-score Histogram')
    plt.show()
```

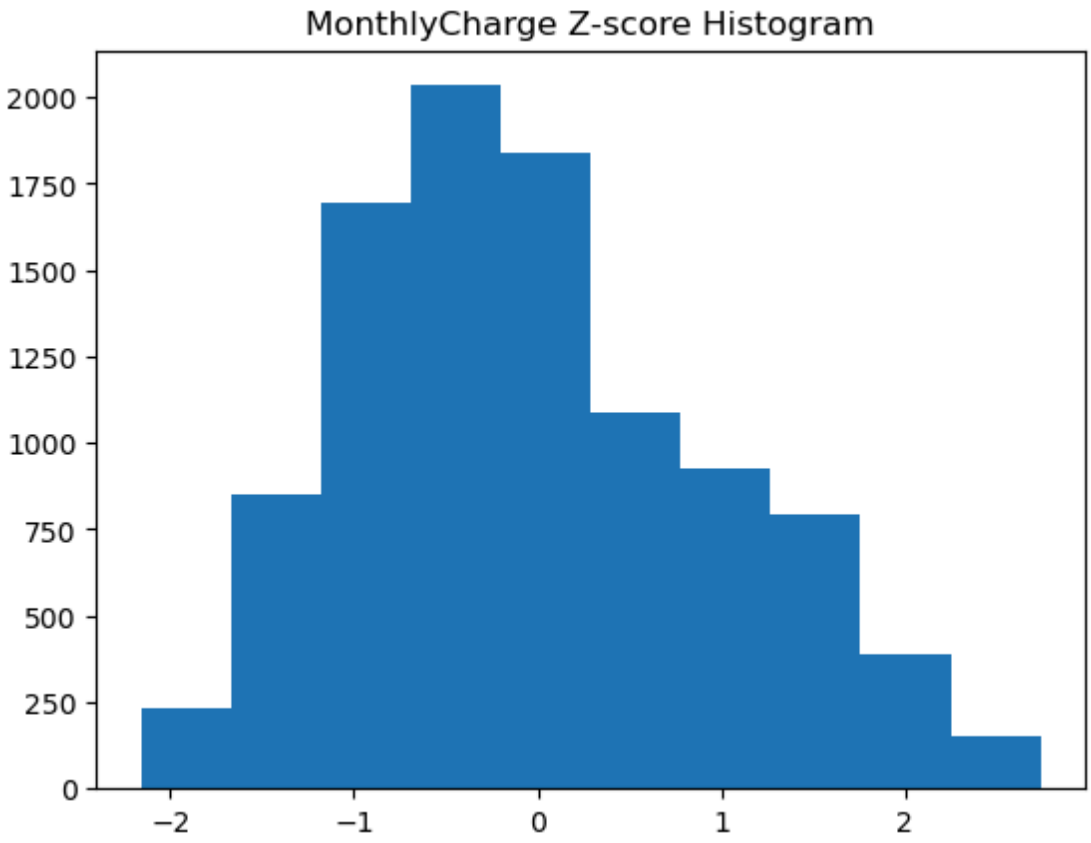


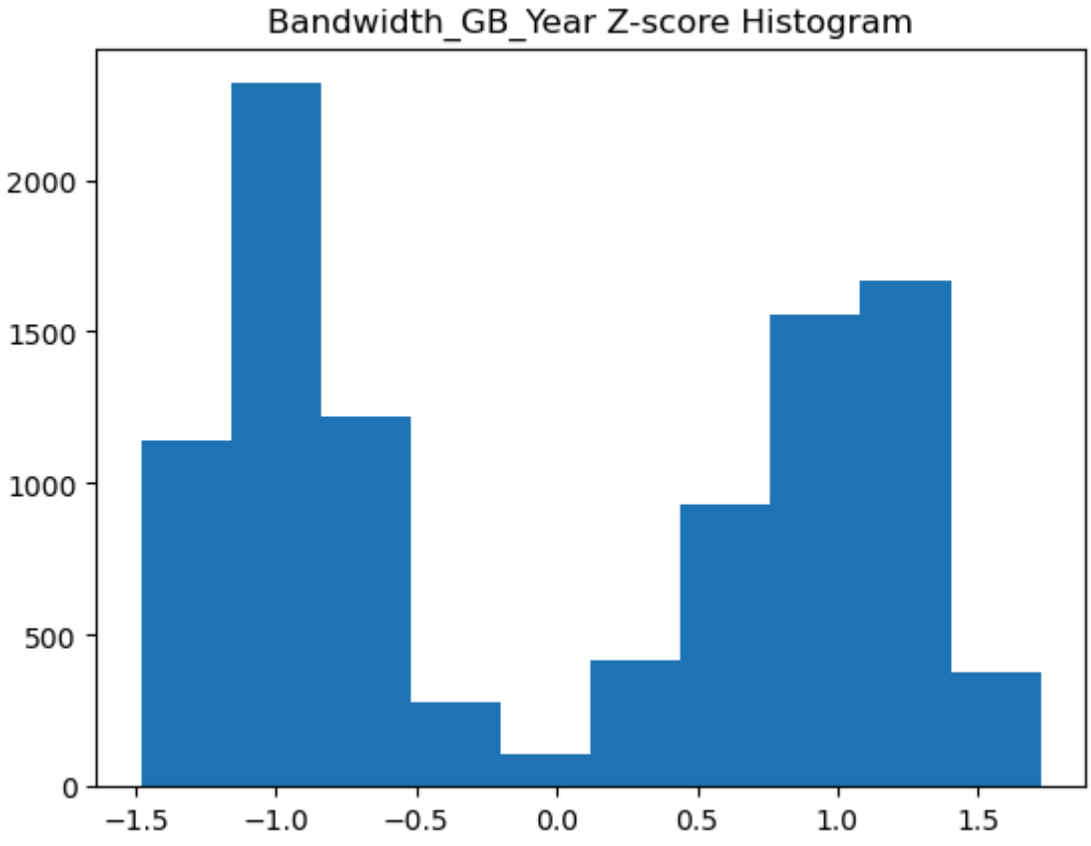

Lng Z-score Histogram
2000
1500
1000
500
0
−2
−1
0
1
2


Income Z-score Histogram
2000
1750
1500
1250
1000
750
500
250
0
−1
0
1
2
3


Outage_sec_perweek Z-score Histogram
2000
1750
1500
1250
1000
750
500
250
0
−2
−1
0
1
2


Tenure Z-score Histogram
2500
2000
1500
1000
500
0
−1.0
−0.5
0.0
0.5
1.0
1.5





In [141…

```
df = df.drop('zscore',axis=1)
```

Normalize Dataset

Next, I will standardize the variables so that they are the same scale and can be used to create principal components for our PCA

In [142…
```
# initialize and fit the standard scaler
scaler = StandardScaler()
norm_df = scaler.fit_transform(df)
```

In [143…
```
# normalize the data
scaled_df = pd.DataFrame(norm_df, columns = df.
```

In [144…
```
scaled_df.head()
```

Out[144]:

	Lat	Lng	Income	Outage_sec_perwee
0	0.128903	0.141679	-0.334208	-0.79520
1	1.207255	0.370086	-0.655793	0.66657
2	1.438670	0.141679	-1.223023	0.29480
3	-1.353068	-2.417619	-0.786145	1.92943
4	-2.161305	-0.606744	0.205681	-0.72877

In [145…
```
scaled_df.to_csv('task2_prepared_data.csv')
```

In [146…
```
scaled_df.shape
```

Out[146]:
```
(10000, 7)
```

D1:PRINCIPAL COMPONENTS

Next we will perform the principal component analysis on the transformed data and then create a loading matrix. Each of the values in a column is the weight which indicates the correlation between the variable in the column and the associated principle component in the row. For instance, I can see that for MonthlyCharge, PC5 has a weight of .689 and PC4 has a weight of -.081. From this, I can understand that principle component 5 is more signficantly related to MonthlyCharge than principle component 4 will be. We can use these weights to identify which of these variables are contributing the most to each principle component.

In [147…
```
pca = PCA()
```

In [148…
```
pc = pca.fit_transform(scaled_df)
```

In [149…
```
loading_matrix = pd.DataFrame(pca.components_,
```

In [150…
```
loading_matrix
```

Out[150]:

	Lat	Lng	Income	Outage_sec_perv
PC1	-0.029524	-0.006876	0.000212	0.008
PC2	0.691645	0.694831	0.156713	0.092
PC3	-0.144046	0.016513	-0.155446	0.669
PC4	-0.079668	-0.162213	0.936169	0.290
PC5	-0.000756	-0.038944	0.261747	-0.673
PC6	0.702608	-0.699329	-0.079693	0.078
PC7	0.000780	0.000154	-0.001492	-0.000

D2:IDENTIFICATION OF THE TOTAL NUMBER OF COMPONENTS

There are two ways to identify the signifcant components of PCA. The first is a scree plot elbow method in which one finds the point in a scree plot where there is a kink in the line. This is the point that identifies the significant prinicples components. However, Kaiser Criterion is more clear in that it plots the eigenvalues of each of the principles components, and any principle component with an eigenvalue over 1 will be considered significant. I have both methods here, but will use Kaiser Criterion for the main part of my analysis.

In [151…
```
exp_var = pca.explained_variance_ratio_
```

In [152…
```
exp_var
```

Out[152]:
```
array([0.28486723, 0.16258194, 0.1453062 , 0.1
4257386, 0.14035539,
       0.12339317, 0.00092221])
```

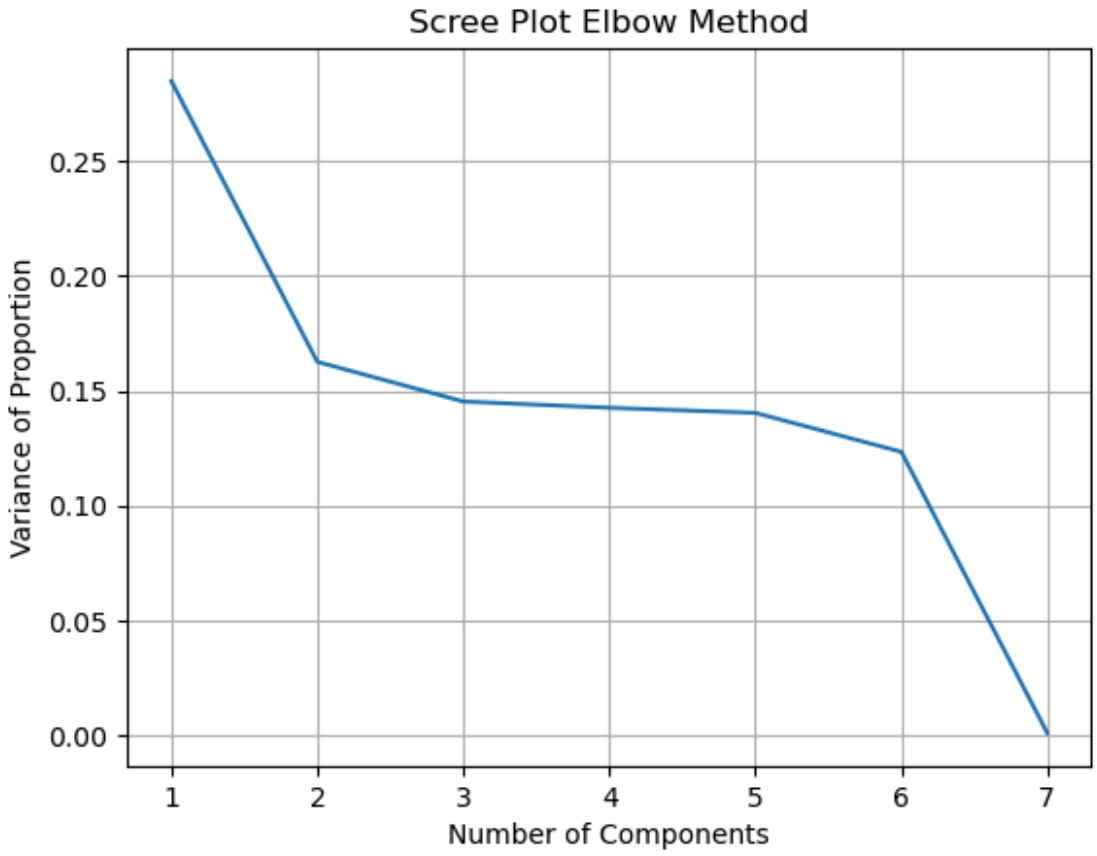
In [153…
```
pcomp = np.arange(pca.n_components_) + 1
```

In [154…
```
pcomp
```

Out[154]:
```
array([1, 2, 3, 4, 5, 6, 7])
```

Scree Plot Elbow Method

In [155…
```
plt.plot(pcomp,
         exp_var)
plt.title('Scree Plot Elbow Method')
plt.xlabel('Number of Components')
plt.ylabel('Variance of Proportion')
plt.grid()
plt.show()
```



Scree Plot Kaiser Criterion

There is another method that is more certain, called the Kaiser Criterion. Anythign over an eigenvalue of 1 will be considered as a usable principle component

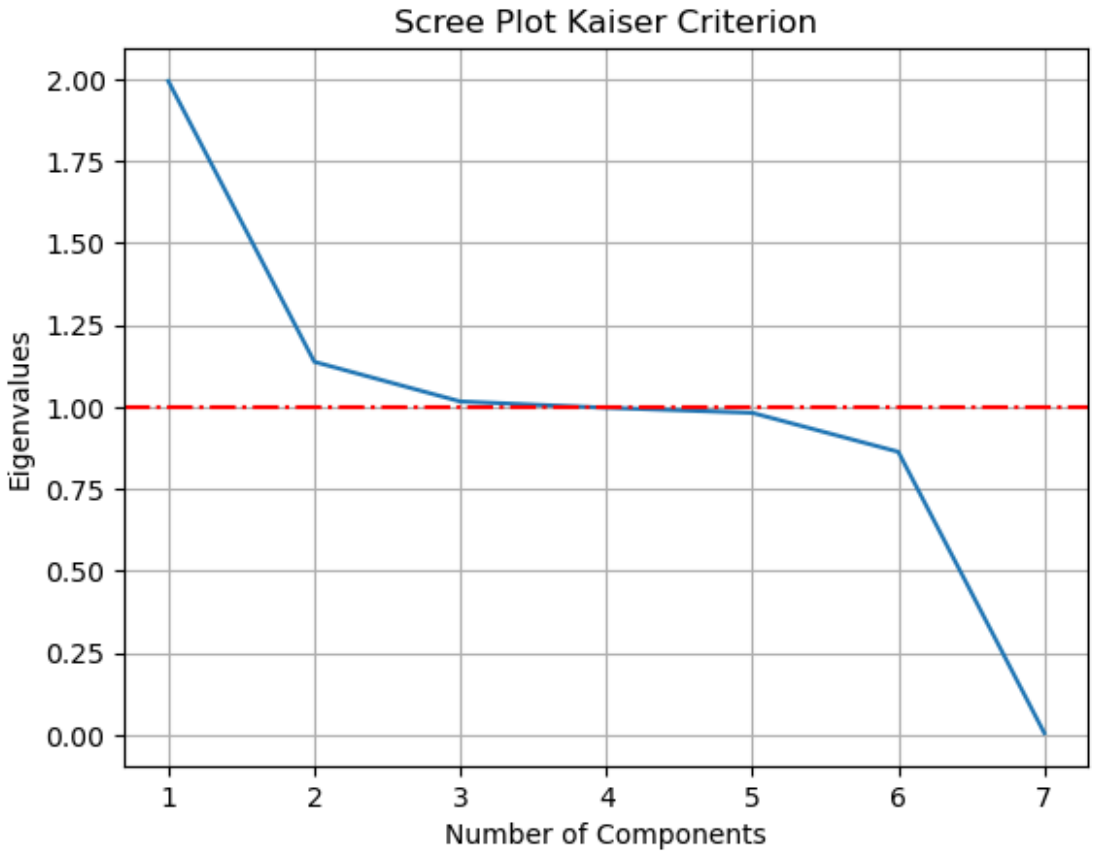
In [156…
```
var = pca.explained_variance_
```

In [157…
```
var
```

Out[157]:
```
array([1.99427003, 1.13818742, 1.01724513, 0.9
9811681, 0.98258602,
       0.86383858, 0.00645609])
```

In [158…
```
plt.plot(pcomp, var)
plt.title('Scree Plot Kaiser Criterion')
plt.xlabel('Number of Components')
plt.ylabel('Eigenvalues')
```

```
plt.axhline(y=1, color = 'r', linestyle = 'dash
plt.grid()
plt.show()
```



We can see from the Kaiser criterion that we should be using up to the first three principle components, as they are above an eigenvalue of 1. The significant comoponents identifed by the Kaiser Criterion are therefore PC1, PC2, and PC3.

In [159…
```
print(dict(zip(['PC1','PC2','PC3'],pcomp)))
```

```
{'PC1': 1, 'PC2': 2, 'PC3': 3}
```

D3:VARIANCE OF EACH COMPONENT

Helps us understand the linear relationship between the variables that make up the principle component.

In [160…
```
print('The variance of the first 3 principle co
print(pca.explained_variance_[:3])
```

```
The variance of the first 3 principle component
s:
[1.99427003 1.13818742 1.01724513]
```

The variance of the first 3 principle components are of the following:

- Principle Component 1 has a variance of 1.99
- Principle Component 2 has a variance of 1.14
- Principle Component 3 has a variance of 1.02

These are the variances of each of the components and it explains variability in the data that is captured by each component

D4:TOTAL VARIANCE CAPTURED BY COMPONENTS

Here we calculate the total variance. Total variance is the total sum of variances that is explainable by all of the primary components. For this PCA, all of the variances add up to 59.28%.

In [161…
```
# total variance
total_variance = np.sum(pca.explained_variance_
total_variance
```

Out[161]: 0.5927553713899526

Total variance captured by the principal components for just specific selected ones:

In [162…
```
total_variance * 100
```

Out[162]: 59.27553713899526

D5:SUMMARY OF DATA ANALYSIS

The total variance of this PCA was 59.28%. This means that PC1, PC2, and PC3 captures 59.28% of the variability of the original dataset. This means that we have identified redundency among the original 7 continuous numerical variables that I started the analysis with. This has been reduced down to 3 significant principle variables using the Kaiser Criterion method. The principle components explained variance was also determined, which helps us understand their own relationships to the original dataset, and how much of the variability in the original dataset each principal component can explain.

We can also connect each principal component to significance in variables in the original dataset. Principal component 1 for instance has high loading in relation to Tenue and Bandwidth_GB_Year. This

means that it is related mostly to length of a customer with the company and their usage. Then principle component 2 is strongly tied to Lat and Lng so it is more of a geographic component. Finally, principal component 3 is most strongly tied to outage seconds per week and monthly charge so it is related to cost and time out of service for a customer. we can also interpret that components that contribute to the same principal component like latitute and longitutde are somewhat redundent.

E:SOURCES FOR THIRD-PARTY CODE

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.a
id=8659c979-9417-432c-acef-
b0bc000f31d5&start=10.788731

F:SOURCES

https://medium.com/@roshmitadey/understanding-
principal-component-analysis-pca-d4bb40e12d33

https://www.youtube.com/watch?
v=HMOI_lkzW08&ab_channel=StatQuestwithJoshStarme

https://www.keboola.com/blog/pca-machine-learning#:~:text=PCA%20assumes%20a%20correlation%