Part One Research Question

A) My question is if customers have a high number of contacts does that lead to a higher churn rate.

B) The two columns I am going to look at for this question is the churn and contacts columns. The churn column is a boolean object showing a 0 or 1. 0 if the customer hasn't left and a 1 if the customer has left. The contacts column has an integer datatype showing the sum of contacts to customer support.

In [2]:
```
import pandas as pd
import numpy as np
from pandas import DataFrame
import scipy.stats as stats
import csv
```

Part Two Data-Cleaning Plan

C1) One step of my data cleaning plan is to look for outliers in my data. I am going to do this in two ways. First I am going to conduct a z-test. Secondly I am going to create a box-plot and histogram. Afer looking for outliers I am going to check for Nan values in each column and then sum the number of nan values for each row.

C2) The data being assessed is the churn column and the contacts column. These columns hold boolean and integer values respectively. My approach for cleaning the data is looking at it in a pragmatic way in order to answer my research question. I will use the z-score to see if there are any outliers in the data set. The z-test can identify outliers if they are higher or lower than what the expected average is.

C3) I am going to be using the python libraries pandas and numpy in my data cleaning. These libaries will allow me to look for outliers, missing data, and sum the amount of missing data.

C4) CLEANING CODE BELOW

In [3]:
```
df = pd.read_csv('churn_raw_data.csv')
```

After loading in my data set I am going to break down the dataframe into a new data frame with my two columns that I want. In this instance I am going to be looking at number of contacts and the churn column.

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 52 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            10000 non-null  int64
 1   CaseOrder             10000 non-null  int64
 2   Customer_id           10000 non-null  object
 3   Interaction           10000 non-null  object
 4   City                  10000 non-null  object
 5   State                 10000 non-null  object
 6   County                10000 non-null  object
```

```
 7   Zip                   10000 non-null  int64
 8   Lat                   10000 non-null  float64
 9   Lng                   10000 non-null  float64
 10  Population            10000 non-null  int64
 11  Area                  10000 non-null  object
 12  Timezone              10000 non-null  object
 13  Job                   10000 non-null  object
 14  Children              7505 non-null   float64
 15  Age                   7525 non-null   float64
 16  Education             10000 non-null  object
 17  Employment            10000 non-null  object
 18  Income                7510 non-null   float64
 19  Marital               10000 non-null  object
 20  Gender                10000 non-null  object
 21  Churn                 10000 non-null  object
 22  Outage_sec_perweek    10000 non-null  float64
 23  Email                 10000 non-null  int64
 24  Contacts              10000 non-null  int64
 25  Yearly_equip_failure  10000 non-null  int64
 26  Techie                7523 non-null   object
 27  Contract              10000 non-null  object
 28  Port_modem            10000 non-null  object
 29  Tablet                10000 non-null  object
 30  InternetService       10000 non-null  object
 31  Phone                 8974 non-null   object
 32  Multiple              10000 non-null  object
 33  OnlineSecurity        10000 non-null  object
 34  OnlineBackup          10000 non-null  object
 35  DeviceProtection      10000 non-null  object
 36  TechSupport           9009 non-null   object
 37  StreamingTV           10000 non-null  object
 38  StreamingMovies       10000 non-null  object
 39  PaperlessBilling      10000 non-null  object
 40  PaymentMethod         10000 non-null  object
 41  Tenure                9069 non-null   float64
 42  MonthlyCharge         10000 non-null  float64
 43  Bandwidth_GB_Year     8979 non-null   float64
 44  item1                 10000 non-null  int64
 45  item2                 10000 non-null  int64
 46  item3                 10000 non-null  int64
 47  item4                 10000 non-null  int64
 48  item5                 10000 non-null  int64
 49  item6                 10000 non-null  int64
 50  item7                 10000 non-null  int64
 51  item8                 10000 non-null  int64
dtypes: float64(9), int64(15), object(28)
memory usage: 4.0+ MB
```

In [5]:
```
df2 = df[['Contacts','Churn']].copy()
```

In [6]:
```
df2
```

Out[6]:

	Contacts	Churn
0	0	No
1	0	Yes
2	0	No
3	2	No
4	2	Yes

	Contacts	Churn
...	...	...
9995	2	No
9996	2	No
9997	0	No
9998	1	No
9999	1	No

10000 rows × 2 columns

z- test for outliers in the contacts column

In [7]:
```
df2['Contacts_Zscore'] = stats.zscore(df2.iloc[::,0])
```

In [8]:
```
df2.head(50)
```

Out[8]:

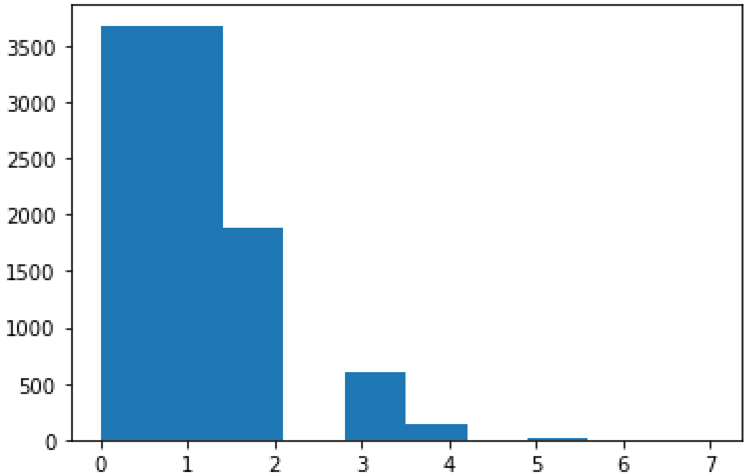
	Contacts	Churn	Contacts_Zscore
0	0	No	-1.005852
1	0	Yes	-1.005852
2	0	No	-1.005852
3	2	No	1.017588
4	2	Yes	1.017588
5	3	No	2.029307
6	0	Yes	-1.005852
7	0	Yes	-1.005852
8	2	No	1.017588
9	1	No	0.005868
10	0	No	-1.005852
11	1	No	0.005868
12	0	No	-1.005852
13	1	No	0.005868
14	3	Yes	2.029307
15	1	Yes	0.005868
16	1	Yes	0.005868
17	3	Yes	2.029307
18	1	No	0.005868
19	1	Yes	0.005868
20	0	No	-1.005852

	Contacts	Churn	Contacts_Zscore
21	1	No	0.005868
22	1	No	0.005868
23	0	No	-1.005852
24	1	Yes	0.005868
25	1	Yes	0.005868
26	0	Yes	-1.005852
27	0	Yes	-1.005852
28	2	Yes	1.017588
29	1	Yes	0.005868
30	0	No	-1.005852
31	2	No	1.017588
32	0	Yes	-1.005852
33	1	Yes	0.005868
34	1	Yes	0.005868
35	0	Yes	-1.005852
36	1	Yes	0.005868
37	2	No	1.017588
38	1	No	0.005868
39	1	No	0.005868
40	1	No	0.005868
41	2	No	1.017588
42	1	Yes	0.005868
43	0	No	-1.005852
44	0	Yes	-1.005852
45	1	Yes	0.005868
46	1	Yes	0.005868
47	1	Yes	0.005868
48	2	No	1.017588
49	0	No	-1.005852

The next sections of code will look for outliers using histograms and boxplots.

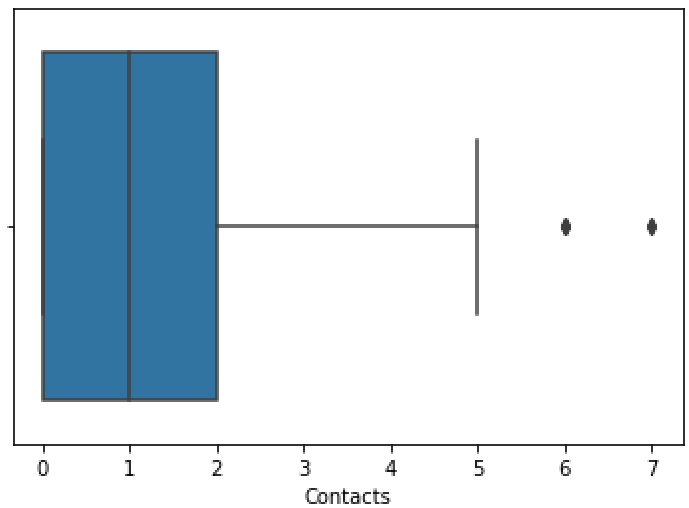
In [9]:
```
import matplotlib.pyplot as plt
```

In [10]:
```
plt.hist(x = 'Contacts', data = df2)
plt.title = 'Contacts'
```



In [11]:
```
import seaborn
```

In [12]:
```
seaborn.boxplot(x='Contacts', data = df2)
plt.title = 'Contacts'
```



The next sections of code will be looking at missing data.

In [13]:
```
df2.isna().sum()
```

Out[13]:
```
Contacts          0
Churn             0
Contacts_Zscore   0
dtype: int64
```

In [14]:
```
df2.isna().sum(axis=1)
```

Out[14]:
```
0       0
1       0
2       0
3       0
4       0
       ..
9995    0
9996    0
9997    0
9998    0
```

```
9999    0
Length: 10000, dtype: int64
```

Part Three Data Cleaning

D1) My findings were that there were no missing or null values in my two columns needed for the research questions. When looking at outliers it looks like the contacts column has a outlier of 5. But not very many customers have 5 contacts so I left it in the data set.

D2) I left the rows with 5 contacts because it was not a large number of rows.

D3) Clean data set below (data set only has two columns because those are the columns I need to answer my research question)

In [15]:
```
df2
```

Out[15]:

	Contacts	Churn	Contacts_Zscore
0	0	No	-1.005852
1	0	Yes	-1.005852
2	0	No	-1.005852
3	2	No	1.017588
4	2	Yes	1.017588
...	...	...	...
9995	2	No	1.017588
9996	2	No	1.017588
9997	0	No	-1.005852
9998	1	No	0.005868
9999	1	No	0.005868

10000 rows × 3 columns

If I didn't have access to all the data or only chunks of the data that would limit what I could do to clean the data. The implications of this would be a report that isn't accurate and could lead to misleading conclusions.

running a principal component analysis on original data set (df)

In [16]:
```
df.dtypes
```

Out[16]:
```
Unnamed: 0             int64
CaseOrder              int64
Customer_id           object
Interaction           object
City                  object
State                 object
County                object
```

```
Zip                       int64
Lat                     float64
Lng                     float64
Population                int64
Area                     object
Timezone                 object
Job                      object
Children                float64
Age                     float64
Education                object
Employment               object
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
```

In [17]:
```
df_pca = df[['CaseOrder','Zip','Population','Contacts','Yearly_equip_failure','item1','
```

In [18]:
```
df2_normalized=(df_pca-df_pca.mean())/df_pca.std()
```

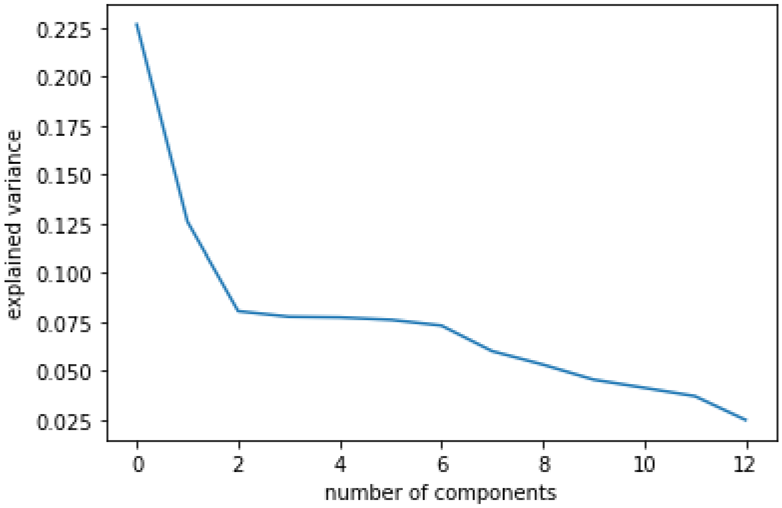
In [19]:
```
from sklearn.decomposition import PCA
```

In [20]:
```
pca = PCA(n_components=df_pca.shape[1])
```

In [21]:
```
pca.fit(df2_normalized)
df_pca2 = pd.DataFrame(pca.transform(df2_normalized),
columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC9','PC10','PC11','PC1
```

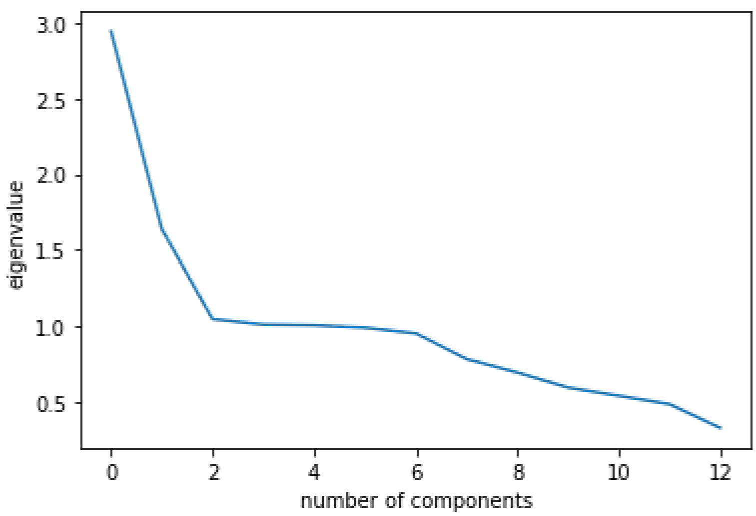
In [22]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [23]:
```
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('number of components')
plt.ylabel('explained variance')
plt.show()
```



In [24]:
```
cov_matrix = np.dot(df2_normalized.T, df2_normalized) / df2.shape[0]
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for eigenvector i
```

In [25]:
```
plt.plot(eigenvalues)
plt.xlabel('number of components')
plt.ylabel('eigenvalue')
plt.show()
```



E1) The principal compoenents in the data set are components 0-2.

E2) I identified these components using two visualiztions. My scree plot tells me that components 0-2 make up more than 50 percent of the data. The components who have an eigen value of above 1 are also components 0-2.

E3) The above PCA results will be beneficial to any company that is looking at the best ways to reduce their data set without losing the integrity of the data itself.

References

Larose, C. D., & Larose, D. T. (2019). Data science using Python and R. John Wiley & Sons. ISBN: 978-1-119-52684-1