EDA and Data Cleaning – Performance Assessment

Steven Schindler

Data Cleaning – D206

Straw, Eric; PhD

College of I.T., Western Governors University

Table of Contents

Part I: Research Question and Variables 3

A. 3

B. 3

Part II: Data-Cleaning Plan (Detection) 7

C1. 7

C2. 8

C3. 9

C4. 9

Part III: Data Cleaning (Treatment) 14

D1. 14

D2. 21

D3. 24

D4. 25

D5. 26

D6. 26

D7. 27

Part IV: PCA 27

E1. 27

E2. 28

E3. 29

Part V: References 30

G. 30

H. 30

Part I: Research Question and Variables

A.

The Data set I am working with is the Telecommunications Churn Data. The data set describes customer account information, customer demographics, churn information and each service a customer can have. My research question is: Which three states have the highest amount of churn? To get the answer one would group the customers by state then add up how many customers have "Yes" in the churn column. Once this information is obtained then sort the states from highest to lowest based on the count then choose the top three.

B.

The variables for the data set are listed as follows:

Unnamed: a numeric datatype that counts how many rows are in the data. E.g., 1,2,3 … ,9998,9999,10000.

CaseOrder: A numeric datatype that also counts how many rows are in the data. E.g., 1,2,3 … ,9998,9999,10000.

Customer_id: A unique identifier that is assigned to each customer it contains both numbers and letters and is a qualitative datatype. E.g., K409198

Interaction: Another unique identifier that is assigned to interactions with the customer such as transactions, tech support and service sign-ups. It is a qualitative datatype with both letters and numbers. E.g., aa90260b-4141-4a24-8e36-b04ce1f4f77b

City: A qualitative string datatype that tells customer city of residence. E.g., Salt Lake City

State: A qualitative string datatype that tells customer state of residence. E.g., Utah

County: A qualitative string datatype that tells customer County of residence. E.g., Prince of Wales-Hyder
Zip: A qualitative numeric datatype that tells the customers zip code. E.g., 84106
Lat: The latitudinal position of the customers residence which is a quantitative numeric datatype. E.g., 56.251

Lng: The longitudinal position of the customers residence which is a quantitative numeric datatype. E.g., -133.37571

Population: A quantitative datatype that counts how many other people live within a mile of the customer. E.g., 38

Area: A qualitative datatype that lists if the customer lives in an urban, suburban or rural area. E.g., Urban

Timezone: A qualitative datatype that lists which time zone the customer is in. E.g., America/New_York

Job: A qualitative datatype that lists the job of the customer. E.g., Solicitor

Children: A quantitative datatype that counts how many children the customer has. E.g., 1,2,3…

Age: A quantitative datatype that lists the age of the customer. E.g., 18, 89…

Education: A qualitative datatype that lists how educated the customer is. E.g., Master's Degree

Employment: A qualitative datatype that lists the customer's employment status. E.g., Part Time, Full Time, etc.

Income: A quantitative datatype that lists how much money a customer makes in a year. E.g., 11,467.50

Marital: A qualitative datatype that lists the marital status of a customer. E.g., Married, Divorced, Single, etc.

Gender: A qualitative datatype that lists the gender of the customer. E.g., Male, Female

Churn: A qualitative datatype that tells if the customer dropped service in the past month. E.g., Yes, no

Outage_sec_perweek: A quantitative datatype that lists the average seconds per week for a customer's neighborhood. E.g., 6.972566093

Email: A quantitative datatype that counts the number of emails sent to the customer within the last year. E.g., 1, 2, 3…

Contacts: A quantitative datatype that counts the number of times the customer contacted tech support. E.g., 1, 2, 3

Yearly_equip_failure: A quantitative datatype that counts the number of times the customers equipment failed and had to be replaced or reset. E.g., 1, 2, 3

Techie: A qualitative datatype that lists if the customer considers themselves technically adept. E.g., Yes, No

Contract: A qualitative datatype that lists the contract term for the customer. E.g., One year, Two year

Port_modem: A qualitative datatype on whether the customer owns a portable modem. E.g., yes, no

Tablet: A qualitative datatype on whether the customer owns a tablet such as an iPad. E.g., yes, no

InternetService: A qualitative datatype on the customers internet provider. E.g., DSL, fiber optic

Phone: A qualitative datatype on if the customer has phone service. E.g., yes, no

Multiple: A qualitative datatype on if the customer has multiple lines or services. E.g., yes, no

OnlineSecurity: A qualitative datatype on if the customer has an online security service. E.g., yes, no

OnlineBackup: A qualitative datatype on if the customer has an online backup service. E.g., yes, no

DeviceProtection: A qualitative datatype on if the customer has device protection. E.g., yes, no
TechSupport: A qualitative datatype on if the customer has the technical support add-on. E.g., yes, no

StreamingTV: A qualitative datatype on if the customer has TV streaming. E.g., yes, no

StreamingMovies: A qualitative datatype on if the customer has movie streaming. E.g., yes, no

PaperlessBilling: A qualitative datatype on if the customer has paperless billing. E.g., yes, no

PaymentMethod: A qualitative datatype on the customers method of payment. E.g., Mailed Check, Credit Card (automatic)

Tenure: A quantitative datatype that lists how many months the customer has been with the company. E.g., 6.79551295, 1.156681

MonthlyCharge: A quantitative datatype that lists the average amount billed to the customer every month. E.g., 171.4497621

Bandwidth_GB_Year: A quantitative datatype that lists the average amount of data, in gigabytes, the customer uses in a year. E.g., 904.5361102

Item 1 – 8: Are quantitative datatypes that list the responses to a questionnaire asking customers to rate the importance of the factors on a scale of 1 to 8 with 1 being the most important and 8 being the least important. E.g., 1,2, … ,7,8. Below are the factors of each item.
Item1: Timely response
Item2: Timely fixes
Item3: Timely replacements
Item4: Reliability
Item5: Options
Item6: Respectful response

Item7: Courteous exchange
Item8: Evidence of active listening

Part II: Data-Cleaning Plan (Detection)

C1.

The plan to assess the quality of data is to detect duplicates, missing values, outliers, non-ascii characters and non – sensical values (E.g., Population = 0). The first thing I used was .head() function from the pandas library to get a preliminary look at the data. There I noticed that the Unnamed column and CaseOrder columns could be equal, so I checked using the .equals() function from pandas and it returned true. So, the unnamed column is redundant and can be dropped.

Next, I use the .replace() function to find and replace all non-ascii characters using the regex expression [^\x00-\x7F]+(regex101. (n.d.).). I replace all non-ascii characters with an empty string to remove them all from the data frame. Then I rounded all numeric values to two decimal places except for age which I rounded to the nearest whole integer. I did this using the .round() function from pandas.

Using the .isna() and .sum() functions and combining them to get .isna().sum() I can see how many NaNs are in each column. From there we see that Children, Age, Income, Techie, Phone, TechSupport, Tenure and Bandwidth_GB_Year all have NaNs.

To find duplicate values I use the .nunique() function that counts how many unique values are in a column I see that for the unique identifiers I get 10,000 each so I now know that there are no duplicates rows. I then use the str.len() and assert function to make sure each state is represented by its initials my making sure the length of the string is no more or no less than two. Then I strip the leading and

trailing spaces on all object datatypes using the str.strip() function. Then finally I check for duplicates using a subset without the unique identifiers to verify I have no duplicate rows using the . duplicated() function.

I look for non sensical values in the quantitative variables by using the max() and min() function of each variable. I discover that population has a min value of zero which cannot be as the customer counts as at least 1. I also discovered that some outages per week have negative values which cannot be. No other quantitative variables have any egregious or non sensical values. I also check whether the categories that have ‘yes’ or ‘no’ values are consistent by printing out their unique values using .unique() function.

To detect outliers, I use a combination of histogram and boxplot charts. Using these methods to locate outliers of every quantitative variable ('Lat', 'Lng', 'Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8'). I found outliers in Lat, Lng, Population, Children, Income, 'Outage_sec_perweek', email, contacts, Yearly_equip_failure, MonthlyCharge and items 1-8.

C2.

The functions that I chose were also the functions used in the DataCamp supplemental exercises. The isna() and sum() functions are good tools to find if any columns have NaNs and how many there are in every column regardless of datatype. While the nunique(), unique() and duplicated() functions are all good functions to count and find unique and duplicated values. I used the replace() function on all datatypes in order to replace all non-ascii characters.

Then using the strip() function I can remove all trailing and leading spaces in the string datatype. This allows me to not have to worry about counting or

detecting trailing and leading spaces as I can just replace them since none are needed. The boxplots and histograms are excellent ways to visualize the data in search for outliers.

C3.

The programming language I used to clean my data was Python. Python has many useful libraires that can be imported to make cleaning data easier. The libraries that I used in my code are the numpy, pandas, string, matplotlib, and the PCA package imported from the Scikit-learn library. I used the panda library help manipulate data sets and the numpy library to aid in computing. The string library is useful for manipulating string datatypes such as using the .strip() method that removes leading and trailing whitespaces. The matplotlib is very useful for graphing data into histograms and boxplots. And finally, the PCA package helps perform PCA calculations. All these useful tools make Python a good choice for cleaning data but, the more subjective reason why I chose to work in Python over other tools is to increase my Python skills.

C4.

The code I used to assess data quality is shown below I have also provided the code in its original format upload as “D206_performance_assessment.ipynb”

```
# import the numpy and pandas libraries
import numpy as np
import pandas as pd
import string as str

#read in the churn data set as a pandas dataframe
df = pd.read_csv('churn_raw_data.csv')
df.info()

#check to see if the first column is identical to CaseOrder
df['Unnamed: 0'].equals(df['CaseOrder'])
```

```
#replace non asci characters
new_df.replace({r'[^\x00-\x7F]+':''}, regex=True, inplace=True)(
```
regex101. (n.d.).*)*

```
# make sure all the quantitative datatypes that need to be rounded are consistent
new_df = new_df.round({'Age':
0,'Tenure':2,'MonthlyCharge':2,'Bandwidth_GB_Year':2,'Outage_sec_perweek':2,'
Income':2}) # round Age to nearest integer

# find the number of NaNs in each column
new_df.isna().sum()

#number of unique rows in each column
new_df[new_df.columns].nunique()
#this verifys all unique identifiers are unique

#make sure the states are consistently length as initials
state_length = new_df['State'].str.len()
assert state_length.max() == 2
assert state_length.min() == 2

#strips leading and trailing spaces
string_list = list(new_df.select_dtypes(include = {'object'}))
for i in string_list:
   new_df[i] = new_df[i].str.strip()

#make sure there are no duplicte rows when the unique identifiers are left out.
duplicates = new_df.duplicated(subset = ['City', 'State', 'County',
      'Zip', 'Lat', 'Lng', 'Population', 'Area', 'Timezone', 'Job',
      'Children', 'Age', 'Education', 'Employment', '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'], keep = False)

new_df[duplicates]
```

```
# making sure the categories with yes/no values are consistent
category_column_name =
['Phone','Multiple','OnlineSecurity','OnlineBackup','DeviceProtection','TechSuppo
rt','StreamingTV','StreamingMovies','PaperlessBilling']

for i in category_column_name:
    print(i,':',new_df[i].unique(),'\n')

# detecting outliers
import matplotlib.pyplot as plt

#Population,Children,Age,Income,Outage_sec_perweek,Email,Contacts,Yearly_eq
uip_failure,Tenure,MonthlyCharge,Bandwidth_GB_Year

col_num_names =
['Lng','Lat','Population','Children','Age','Income','Outage_sec_perweek','Email','C
ontacts','Yearly_equip_failure','Tenure','MonthlyCharge','Bandwidth_GB_Year']

for i in col_num_names:
    print(new_df[i].name,"min is ", new_df[i].min(),"max is ",new_df[i].max(),"\n")

plt.hist(new_df['Population'])
plt.title('Population histogram')

plt.boxplot(new_df['Population'])
plt.title('Population boxplot')

plt.hist(new_df['Lat'])
plt.title('lat histogram')

plt.hist(new_df['Lng'])
plt.title('lng histogram')

plt.boxplot(new_df['Children'])
plt.title('Children boxplot')

plt.hist(new_df['Age'])
plt.title('Age Histogram')
```

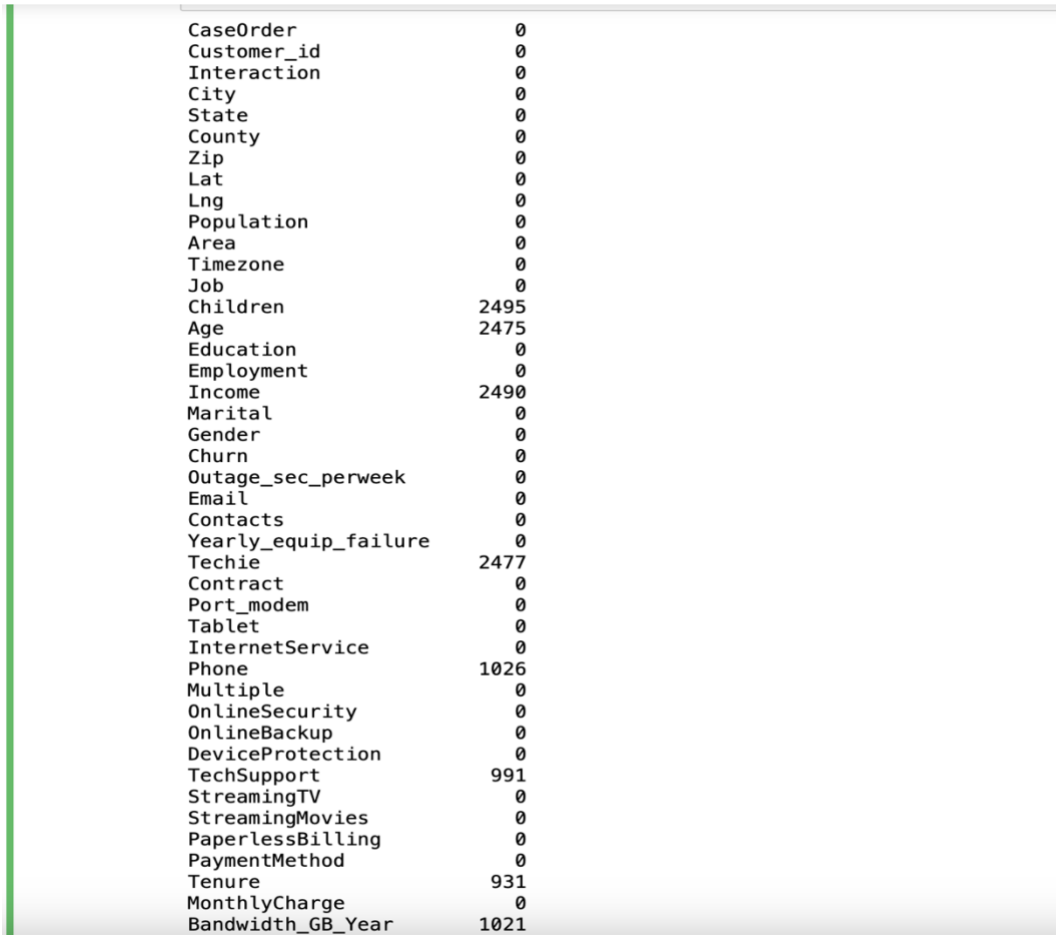
```
plt.boxplot(new_df['Age'])
plt.title('Age boxplot')

plt.boxplot(new_df['Income'])
plt.title('Income boxplot')

plt.hist(new_df['Income'])
plt.title('Income histogram')

plt.hist(new_df['Outage_sec_perweek'])
plt.title('Outage_sec_perweek Histogram')

# from the min max we can see that an Outage_sec_perweek cannot be negative
and from the histogram above
new_df['Outage_sec_perweek']=np.absolute(new_df['Outage_sec_perweek'])

plt.boxplot(new_df['Email'])
plt.title('Email_boxplot')

plt.boxplot(new_df['Contacts'])
plt.title('Contacts_boxplot')

plt.boxplot(new_df['Yearly_equip_failure'])
plt.title('Yearly_equip_failure_boxplot'
plt.hist(new_df['Yearly_equip_failure'])
plt.title('Yearly_equip_failure_histogram')

plt.boxplot(new_df['Tenure'])
plt.title('Tenure')
plt.hist(new_df['Tenure'])
plt.title('Tenure')

plt.hist(new_df['Bandwidth_GB_Year'])
plt.title('Bandwidth_GB_Year')

plt.boxplot(new_df['Bandwidth_GB_Year'])
plt.title('Bandwidth_GB_Year')

plt.hist(new_df['MonthlyCharge'])
plt.title('MonthlyCharge')
```

```
plt.boxplot(new_df['MonthlyCharge'])
plt.title('MonthlyCharge')

plt.boxplot(new_df['item1'])
plt.title('item1')

plt.boxplot(new_df['item2'])
plt.title('item2')

plt.boxplot(new_df['item3'])
plt.title('item3')

plt.boxplot(new_df['item4'])
plt.title('item4')

plt.boxplot(new_df['item5'])
plt.title('item5')

plt.boxplot(new_df['item6'])
plt.title('item6')

plt.boxplot(new_df['item7'])
plt.title('item7')


plt.boxplot(new_df['item8'])
plt.title('item8')
```

Part III: Data Cleaning (Treatment)
D1.

After running the code, I found that the unnamed column was the only duplicate column as it matches the CaseOrder column. I found that Children, Age, Income, Techie, Phone, TechSupport, Tenure and Bandwidth_GB_Year all have missing values. The screenshots below are a list of the number of NaNs in each column. Next, I found 18 columns with outliers I have outlined them in a table below that lists the column name, how many outliers ,what their values were and what tool I used. I have also posted all the graphs I used below the table of outliers. I don't include negative values for the outages per week or zero value for population in the table but those are also outliers as those values cannot be negative or zero respectively.

Number of Missing values:

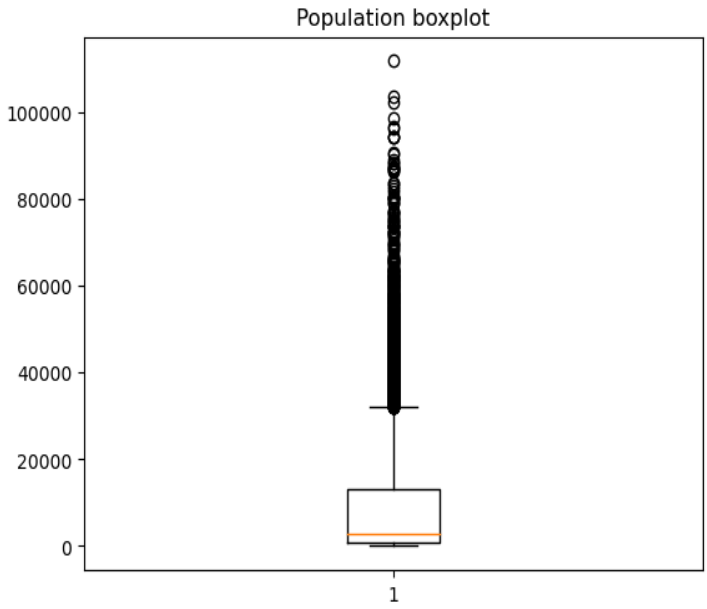


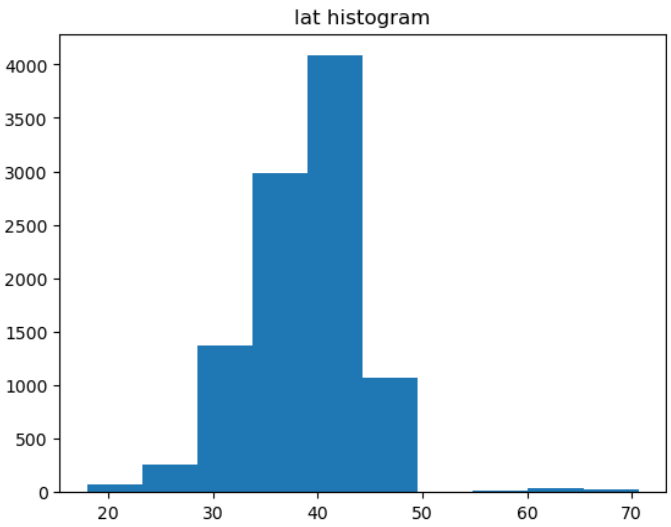
CaseOrder	0
Customer_id	0
Interaction	0
City	0
State	0
County	0
Zip	0
Lat	0
Lng	0
Population	0
Area	0
Timezone	0
Job	0
Children	2495
Age	2475
Education	0
Employment	0
Income	2490
Marital	0
Gender	0
Churn	0
Outage_sec_perweek	0
Email	0
Contacts	0
Yearly_equip_failure	0
Techie	2477
Contract	0
Port_modem	0
Tablet	0
InternetService	0
Phone	1026
Multiple	0
OnlineSecurity	0
OnlineBackup	0
DeviceProtection	0
TechSupport	991
StreamingTV	0
StreamingMovies	0
PaperlessBilling	0
PaymentMethod	0
Tenure	931
MonthlyCharge	0
Bandwidth_GB_Year	1021

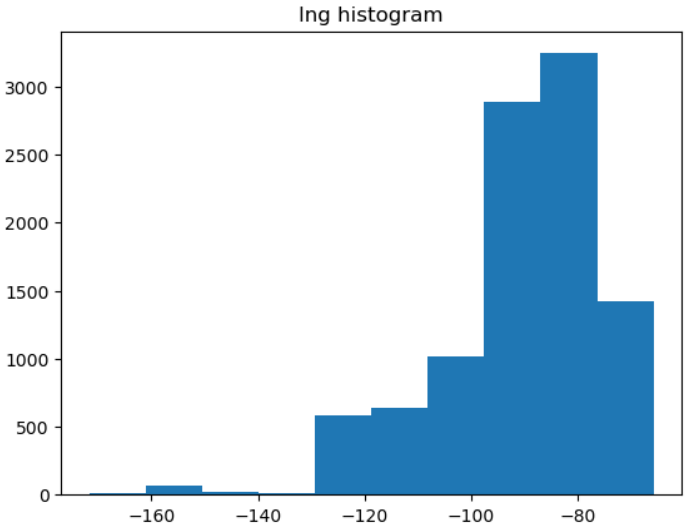
Number of Missing Values Con:

```
Bandwidth_GB_Year       1021
item1                      0
item2                      0
item3                      0
item4                      0
item5                      0
item6                      0
item7                      0
item8                      0
dtype: int64
```

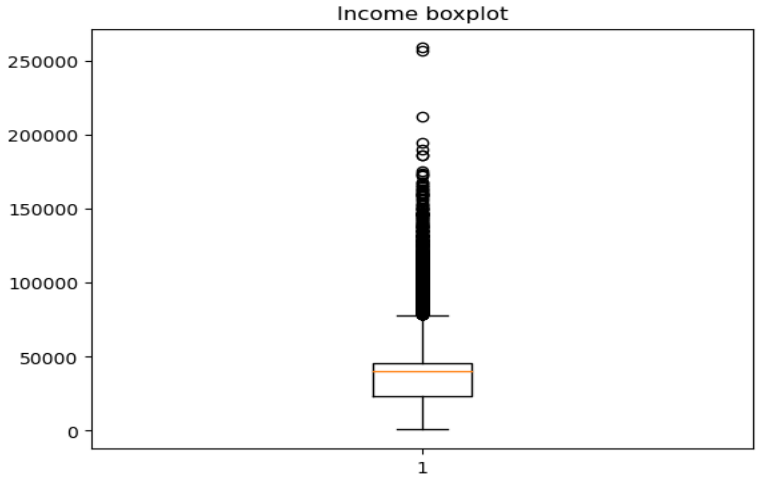
Column Name	Number of Outliers	Value of Outlier	Tool Used
Population	6	Above 90,000	boxplot
Lat	Clustered group	between 55 and 71	histogram
Lng	Clustered group	between -170 and -140	histogram
Children	3	8,9,10	boxplot
Income	6	Above 175,000	boxplot
Outage_sec_perweek	Clustered group	between 33 and 45	histogram
email	6	1,2,3,21,22,23	boxplot
Contacts	2	6,7	boxplot
Yearly_equip_failure	3	3,4,6	boxplot
MonthlyCharge	Clustered group	Above 300	boxplot
item1	3	1,6,7	boxplot
item2	3	1,6,7	boxplot
item3	4	1,6,7,8	boxplot
item4	3	1,6,7	boxplot
item5	3	1,6,7	boxplot
item6	4	1,6,7,8	boxplot
item7	3	1,6,7	boxplot
item8	4	1,6,7,8	boxplot

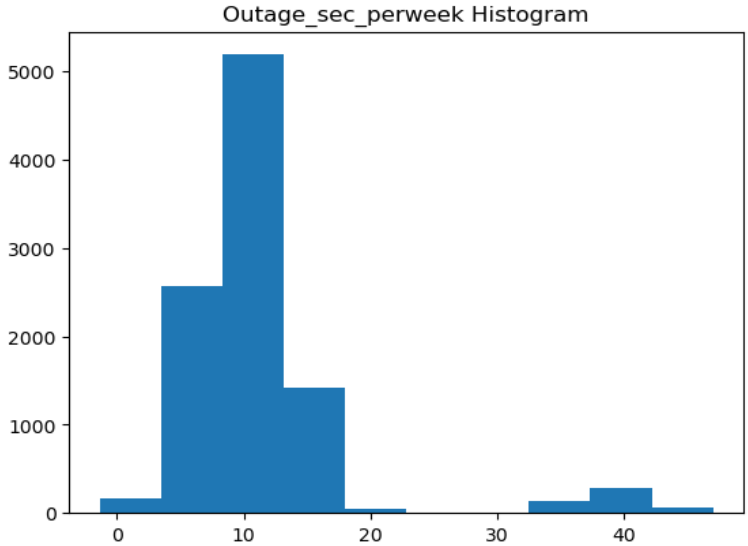

Population boxplot
100000
80000
60000
40000
20000
0
1

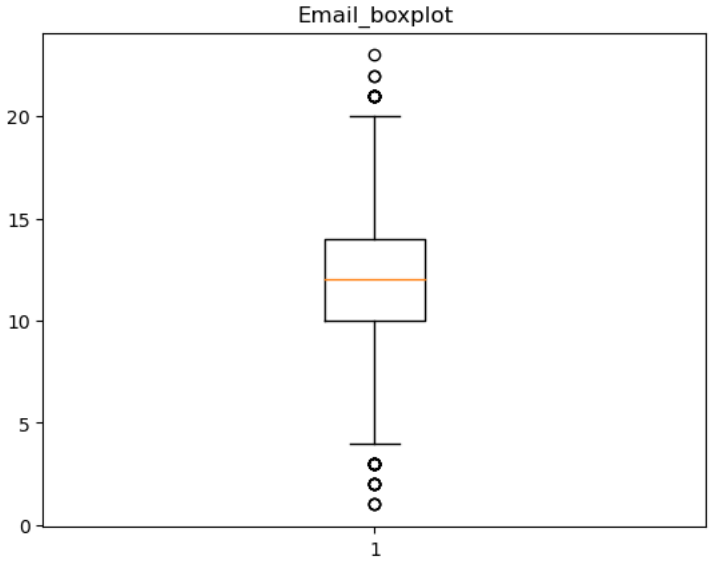

lat histogram
4000
3500
3000
2500
2000
1500
1000
500
0
20
30
40
50
60
70

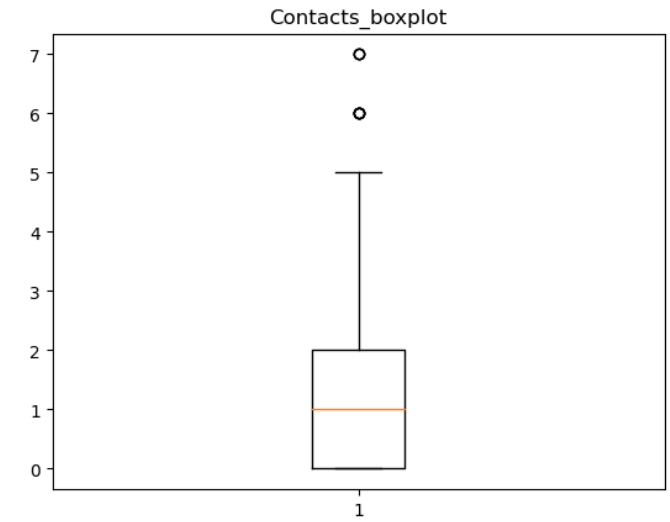

lng histogram
3000
2500
2000
1500
1000
500
0
−160
−140
−120
−100
−80

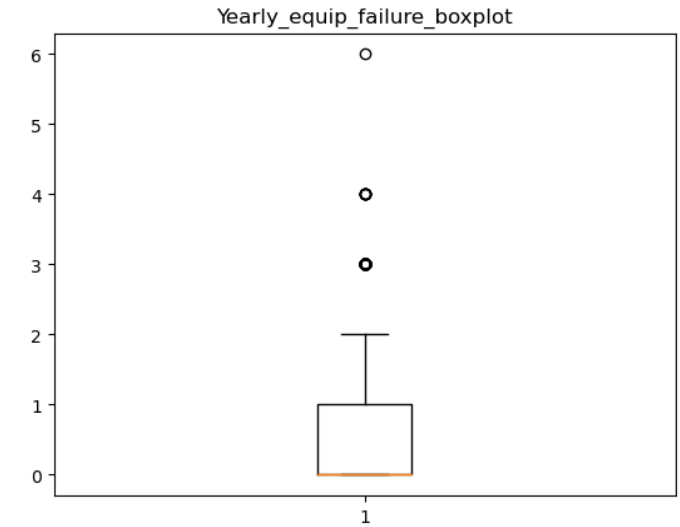

Children boxplot
10
8
6
4
2
0
1

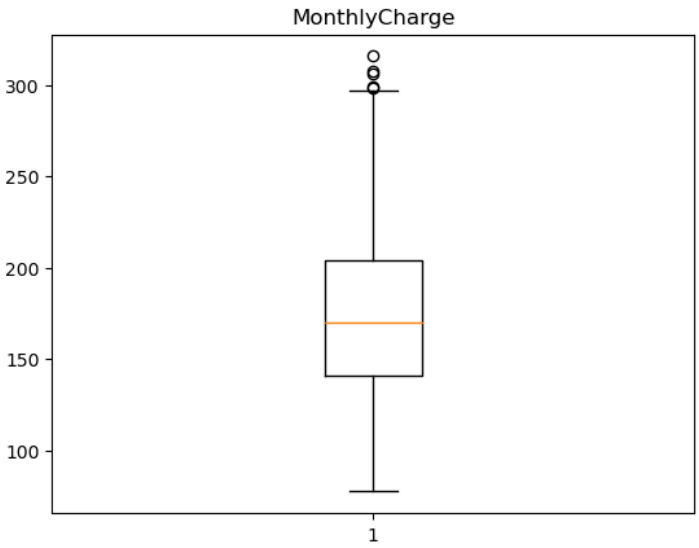

Income boxplot
250000
200000
150000
100000
50000
0
1

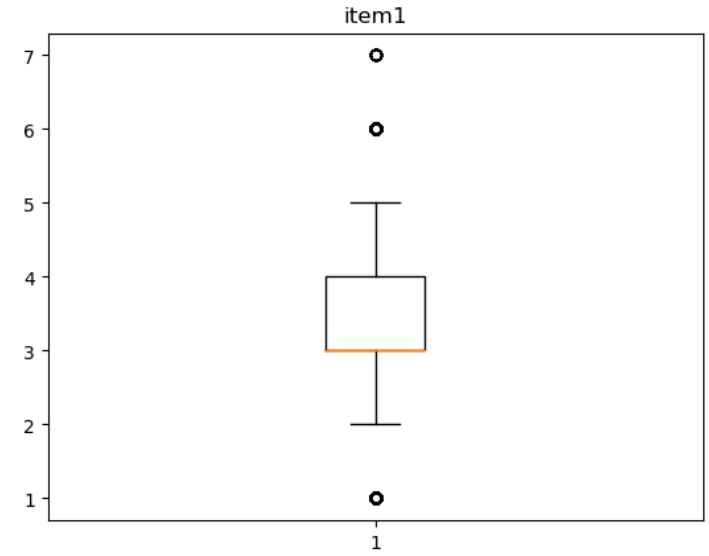

Outage_sec_perweek Histogram
5000
4000
3000
2000
1000
0
0
10
20
30
40

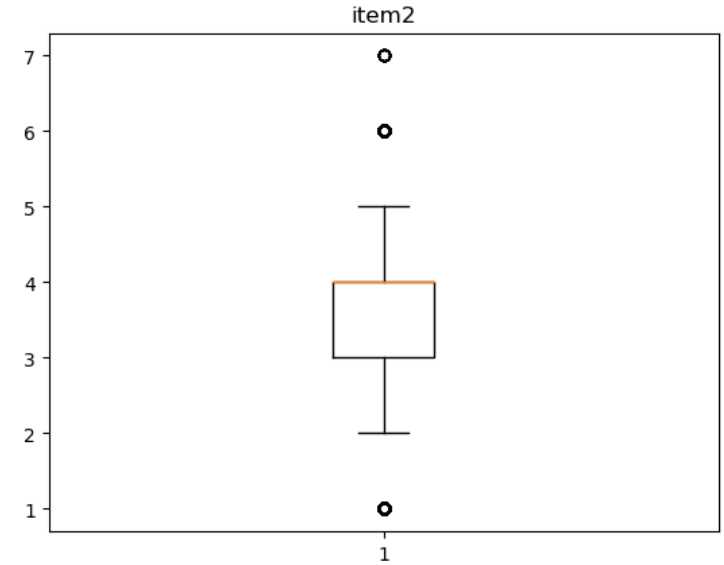

Email_boxplot
20
15
10
5
0
1

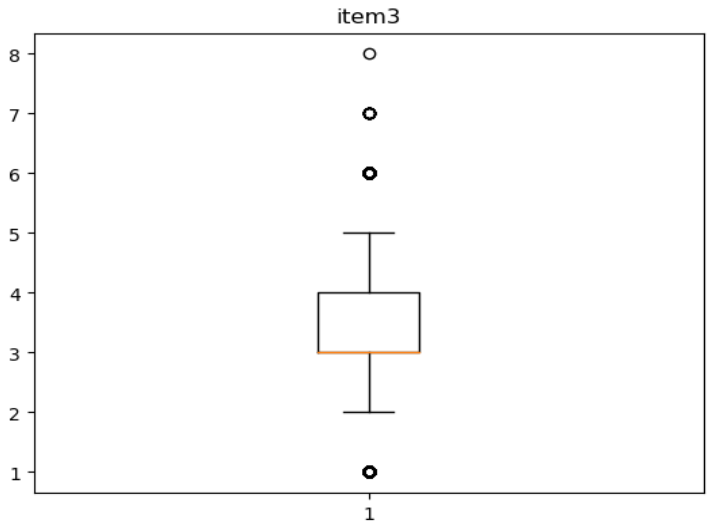

Contacts_boxplot
7
6
5
4
3
2
1
0
1

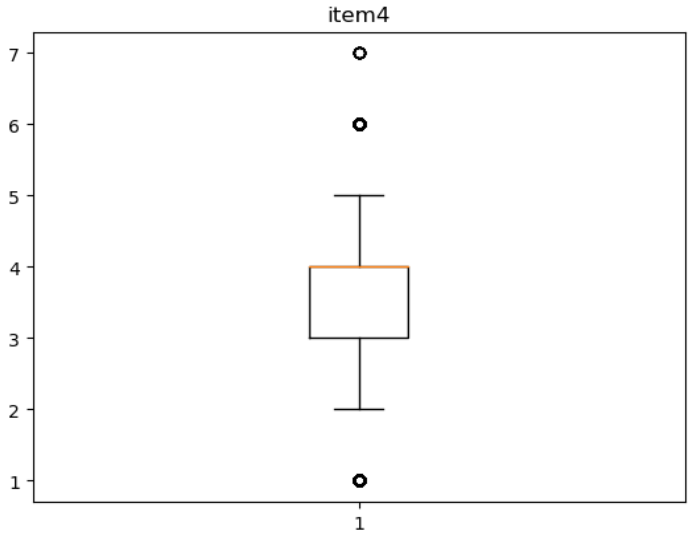

Yearly_equip_failure_boxplot
6
5
4
3
2
1
0
1

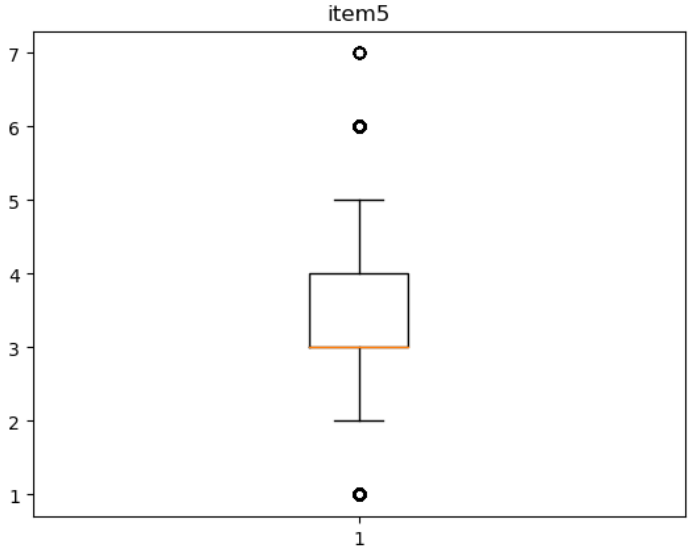

MonthlyCharge
300
250
200
150
100
1

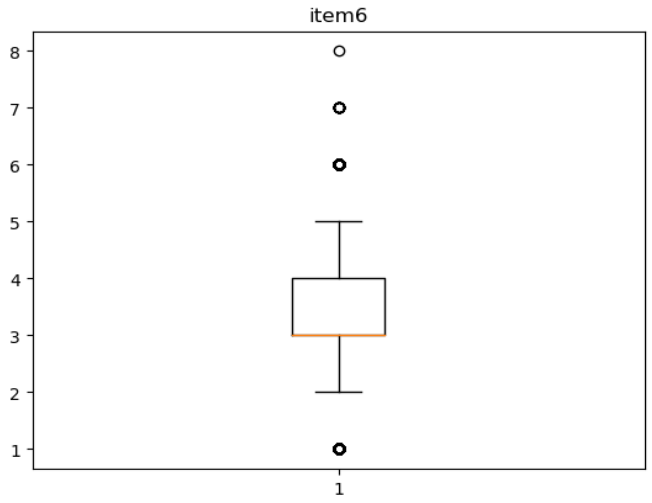

item1
7
6
5
4
3
2
1
1

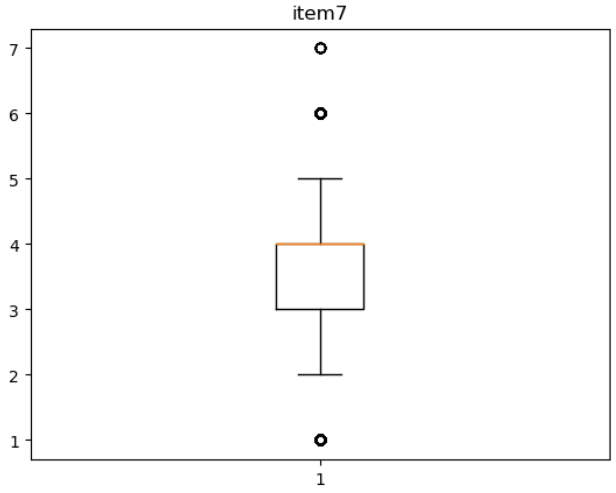

item2
7
6
5
4
3
2
1
1

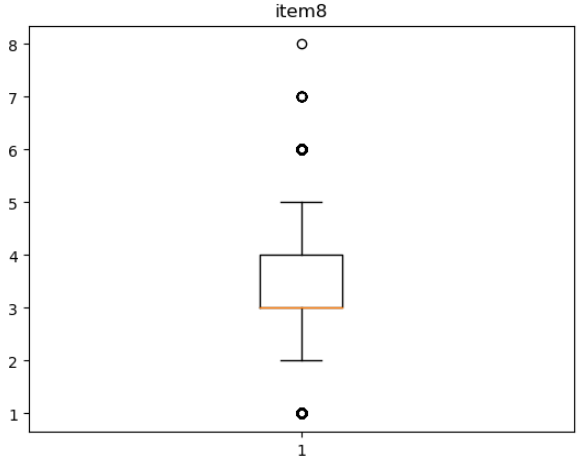

item3
8
7
6
5
4
3
2
1
1


item4
7
6
5
4
3
2
1
1


item5
7
6
5
4
3
2
1
1


item6
8
7
6
5
4
3
2
1
1


item7
7
6
5
4
3
2
1
1


item8
8
7
6
5
4
3
2
1
1

D2.

To deal with the missing values I used several impute methods that were included in the supplemental DataCamp exercises. For the Techie and TechSupport columns I imputed them with their mode which was “no”. Phone and Children I imputed with constant values of ‘no’ and ‘0’ respectively. The reason I chose constants with these two is based on the assumption that a false negative may be better than a false positive in this case.

Whether it is or not depends on what a data analyst is looking for, either way the company should be contacted for the correct responses for Techie, TechSupport and Phone. The Children column has ‘0’ imputed for the blanks to not give children to a customer that does not have any and, in order to preserve the mean of children.

I then imputed Age, Income, Tenure and Bandwidth_GB_Year with the average as imputing the mean will preserve the mean of their overall data respectively. After imputing this data, I have no more missing values as shown by the screenshots below.

To deal with duplicates I just dropped the unneeded unnamed column that was a duplicate to CaseOrder using .drop() method in pandas. The strip method was used earlier in section C2 to clean data of trailing and leading spaces. The same with non-ascii characters were already clean using regex in section C2.

For outliers I kept most of them in as the values in the table are justifiable and is legitimate data. The two that I did change was I removed the records where population was zero as you cannot have a zero-population area for customers. The data removal was 0.97% less than 1% of the data so I felt justified removing it. The second outlier that I removed was in the Outages_sec_perweek column. The outlier had negative values which cannot happen when dealing with time

calculating outages per week. I assume this data was made in error, so I took the absolute value of the whole column to fix negative values.

No more missing values

```
new_df.isna().sum()
#now we have no more missing values
```

```
CaseOrder              0
Customer_id            0
Interaction            0
City                   0
State                  0
County                 0
Zip                    0
Lat                    0
Lng                    0
Population             0
Area                   0
Timezone               0
Job                    0
Children               0
Age                    0
Education              0
Employment             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
```

```
PaymentMethod          0
Tenure                 0
MonthlyCharge          0
Bandwidth_GB_Year      0
item1                  0
item2                  0
item3                  0
item4                  0
item5                  0
item6                  0
item7                  0
item8                  0
dtype: int64
```

D3.

To summarize I've now cleaned every missing value using the technique of imputation. Then I treated duplicates by dropping the unnamed column as well as removing all trailing and leading spaces in each string datatype column. I then replaced all non-ascii characters with blanks. Finally, I removed non-sensical outliers such as Population being '0' and taking the absolute value for Outages_sec_perweek in order to get rid of negative values. I located many more outliers but did not remove them because they are legitimate data.

Below are screenshots of before and after dropping the unnamed column. I have screenshots of before and after of removing the NaNs in section D1 and D2 respectively. Then I also have tables and graphs of the outliers I found in section D1. Then below the two picture I have another screen shot showing the minimum value of Population and Outages_sec_perweek.

2]:

	Unnamed: 0	CaseOrder	Customer_id
0	1	1	K409198
1	2	2	S120509
2	3	3	K191035
3	4	4	D90850
4	5	5	K662701

]:

	CaseOrder	Customer_id
0	1	K409198
1	2	S120509
2	3	K191035
3	4	D90850
4	5	K662701

```
60]: print (new_df['Population'].name,"min is ", new_df['Population'].min())
     print (new_df['Outage_sec_perweek'].name,"min is ", new_df['Outage_sec_perweek'].min())

     Population min is  0
     Outage_sec_perweek min is  -1.348571
```

D4.

The code I used to treat data quality is shown below I have also provided the code in its original format upload as “D206_performance_assessment.ipynb”

```
#it returns true so we can drop the first unnamed column
new_df = df.drop(['Unnamed: 0'], axis=1)

#replace non asci characters
new_df.replace({r'[^\x00-\x7F]+':''}, regex=True, inplace=True)

#strips leading and trailing spaces
string_list = list(new_df.select_dtypes(include = {'object'}))
for i in string_list:
    new_df[i] = new_df[i].str.strip()
#impute the Techie, and TechSupport columns with the mode of No as they are
categorical assuming if it's blank than it is a no
str_mode = max(list(new_df['Techie']),key=list(new_df['Techie']).count)
new_df['Techie'].fillna(str_mode, inplace = True)

str_mode =
max(list(new_df['TechSupport']),key=list(new_df['TechSupport']).count)
new_df['TechSupport'].fillna(str_mode, inplace = True)
```

```
# impute phone with No
new_df['Phone'].fillna('No',inplace = True)

# impute age,income and tenure, and Bandwidth_GB_Year with the average
new_df['Age'].fillna(new_df['Age'].mean(), inplace = True)
new_df['Income'].fillna(new_df['Income'].mean(), inplace = True)
new_df['Tenure'].fillna(new_df['Tenure'].mean(), inplace = True)
new_df['Bandwidth_GB_Year'].fillna(new_df['Bandwidth_GB_Year'].mean(),
inplace = True)

#impute number of Children with 0
new_df['Children'].fillna(0,inplace = True)
new_df.isna().sum()
#now we have no more missing values

# from the min max we can see that an Outage_sec_perweek has negative values
which cannot be
new_df['Outage_sec_perweek']=np.absolute(new_df['Outage_sec_perweek'])
```

D5.

My datafile is called “clean_churn_file.csv” and is included with the submitted files. I used to_csv() to extract it to a csv file.

D6.

The biggest limitation of my data cleaning process was that I did not remove outliers as I found that most of them to be legitimate data. I removed the non sensical outliers of zero for population and negative values for outage_sec_perweek but all other outliers were obtained through legitimate means. An example being the children outlier of 10 while not very common has happened in the past. All the other outliers can be explained intuitively as a reason for why they should be kept. The imputation process while, getting rid of missing values will lead to inaccurate data for individual customers. It may also lead to inaccuracies in the data analysis life cycle in later stages.

D7.

If another data analyst were to use my cleaned data for analysis the biggest challenge, they would face would be my lack of removing most outliers. As I said before most of the outliers are legitimate data that should only be removed if there is good reason. I do not believe that my cleaning would have any negative impact on the research question in part A. An analyst could still count the number of yeses for churn and group them by state without issue.

Another issue that a data analyst may face is that I did remove some data where the population is zero. The data removed was less than 1% so the impact should be minimal. Further issues could be from the imputation of data from missing values. The imputation of Techie, TechSupport, and Phone could lead to inaccuracies for the data analyst if not corrected. The numerical values imputed with the average will preserve the mean of the data but may lead to inaccuracies in other forms of analysis later.

Part IV: PCA

E1.

The variables that I used for the PCA are all the numeric variables: 'Lat', 'Lng', 'Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8.'

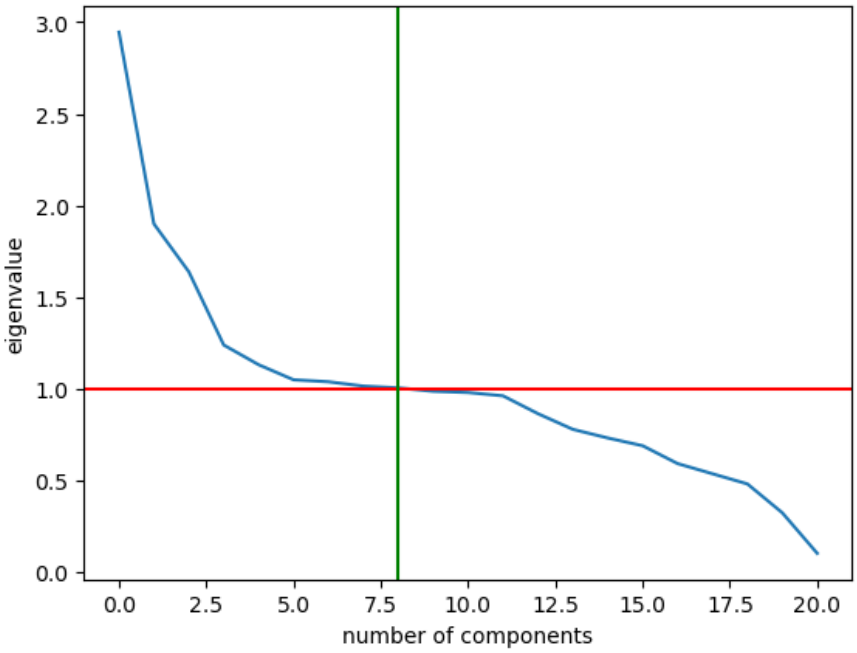
Out[52]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	...	PC12	PC13	
Lat	-0.002743	-0.023845	-0.012798	-0.709381	-0.082707	0.132650	-0.012403	-0.071136	0.002831	0.091843	...	-0.016531	-0.007769	-0.
Lng	0.006610	0.011489	0.025363	0.175537	0.068258	-0.744696	-0.387639	0.028456	-0.050098	-0.258703	...	0.192201	0.058841	-0.
Population	-0.001339	0.004471	0.019963	0.644486	0.059524	0.302944	0.241113	0.049064	0.089674	0.026132	...	-0.137098	-0.005949	-0.
Children	-0.000351	-0.001836	0.011201	-0.045143	0.042804	-0.516366	0.343090	0.016814	0.121360	0.414926	...	-0.615264	0.030157	0.
Age	0.005084	-0.012958	-0.017013	0.020782	-0.053485	0.115897	-0.451734	0.514720	-0.124552	0.340916	...	-0.308130	0.099995	-0.
Income	-0.001695	0.004590	0.024272	-0.058867	-0.025427	-0.092197	0.160589	0.326192	0.813582	0.127152	...	0.365864	-0.072003	0.
Outage_sec_perweek	-0.013389	0.017755	-0.045899	-0.101813	0.692163	0.069588	0.065261	0.035041	0.035175	-0.009569	...	0.066022	0.690736	-0.
Email	0.007398	-0.020356	-0.003043	0.152570	0.104344	0.001214	-0.264068	-0.460741	-0.017828	0.743886	...	0.347683	-0.044305	0.
Contacts	-0.008799	0.005077	-0.010909	0.032240	-0.006353	0.168602	-0.505665	0.268712	0.233107	-0.000349	...	-0.117209	0.019265	0.
Yearly_equip_failure	-0.008617	0.015935	0.007445	-0.020578	0.049673	-0.095083	0.324211	0.575115	-0.485935	0.235581	...	0.424003	-0.114908	0.
Tenure	-0.011137	0.700982	-0.069667	-0.003031	-0.059291	0.009528	-0.012717	-0.009792	-0.002763	0.022199	...	0.006682	0.037938	-0.
MonthlyCharge	-0.000175	0.045380	-0.021544	-0.076325	0.692061	0.035826	-0.088752	0.026823	0.017526	-0.075048	...	-0.128162	-0.684645	0.
Bandwidth_GB_Year	-0.012753	0.702740	-0.071834	-0.009730	-0.010665	-0.001137	0.004384	-0.016344	0.003894	0.011198	...	-0.009659	-0.010433	0.
item1	0.458842	0.031449	0.280643	-0.018488	0.028150	-0.007525	0.001499	0.008175	-0.015603	-0.003434	...	0.002100	-0.005372	-0.
item2	0.434542	0.041957	0.280519	-0.026758	0.011503	0.022390	-0.011540	0.009724	0.000671	0.001564	...	-0.013006	0.001590	-0.
item3	0.400958	0.033140	0.280267	-0.003532	-0.010060	0.000809	-0.004121	-0.033235	-0.026675	-0.002297	...	0.014358	-0.010673	-0.
item4	0.145487	-0.049285	-0.567264	0.004272	-0.028825	-0.012821	-0.005048	-0.006544	-0.019363	-0.020765	...	0.014547	-0.018891	-0.
item5	-0.175312	0.064207	0.585621	-0.015300	0.020694	0.039733	-0.023150	0.005373	-0.007718	-0.005049	...	-0.012031	0.040890	0.
item6	0.404773	-0.010541	-0.184409	0.013941	0.008157	0.015234	0.012613	0.009491	0.004988	0.021117	...	-0.001152	-0.007838	-0.
item7	0.357947	-0.002153	-0.181951	-0.016042	-0.033028	-0.018987	0.000998	0.004748	0.058498	0.024465	...	0.009358	-0.027924	-0.
item8	0.308623	-0.016854	-0.131029	0.043098	0.033368	0.030167	0.012322	0.033087	-0.009489	-0.051696	...	-0.022173	0.119968	0.

21 rows × 21 columns

E2.

Only the first nine PCs should be kept as demonstrated by the scree plot below. The first nine PCs are the only ones with eigenvalues above 1 so those are the PCs that should remain.



E3.

The benefit of the PCA is now we are able to group the 21 variables into 9 more manageable groups. The two tables below highlight which variables have the strongest correlated coefficients and may have insight into the groups that can be created.

	PC1	PC2	PC3	PC4	PC5	PC6
Lat	-0.002743	-0.023845	-0.012798	-0.709381	-0.082707	0.13265
Lng	0.00661	0.011489	0.025363	0.175537	0.068258	-0.744696
Population	-0.001339	0.004471	0.019963	0.644486	0.059524	0.302944
Children	-0.000351	-0.001836	0.011201	-0.045143	0.042804	-0.516366
Age	0.005084	-0.012958	-0.017013	0.020782	-0.053485	0.115897
Income	-0.001695	0.00459	0.024272	-0.058867	-0.025427	-0.092197
Outage_sec_perweek	-0.013389	0.017755	-0.045899	-0.101813	0.692163	0.069588
Email	0.007398	-0.020356	-0.003043	0.15257	0.104344	0.001214
Contacts	-0.008799	0.005077	-0.010909	0.03224	-0.006353	0.168602
Yearly_equip_failure	-0.008617	0.015935	0.007445	-0.020578	0.049673	-0.095083
Tenure	-0.011137	0.700982	-0.069667	-0.003031	-0.059291	0.009528
MonthlyCharge	-0.000175	0.04538	-0.021544	-0.076325	0.692061	0.035826
Bandwidth_GB_Year	-0.012753	0.70274	-0.071834	-0.00973	-0.010665	-0.001137
item1	0.458842	0.031449	0.280643	-0.018488	0.02815	-0.007525
item2	0.434542	0.041957	0.280519	-0.026758	0.011503	0.02239
item3	0.400958	0.03314	0.280267	-0.003532	-0.01006	0.000809
item4	0.145487	-0.049285	-0.567264	0.004272	-0.028825	-0.012821
item5	-0.175312	0.064207	0.585621	-0.0153	0.020694	0.039733
item6	0.404773	-0.010541	-0.184409	0.013941	0.008157	0.015234
item7	0.357947	-0.002153	-0.181951	-0.016042	-0.033028	-0.018987
item8	0.308623	-0.016854	-0.131029	0.043098	0.033368	0.030167

	PC7	PC8	PC9
Lat	-0.012403	-0.071136	0.002831
Lng	-0.387639	0.028456	-0.050098
Population	0.241113	0.049064	0.089674

Children	0.34309	0.016814	0.12136
Age	-0.451734	0.51472	-0.124552
Income	0.160589	0.326192	0.813582
Outage_sec_perweek	0.065261	0.035041	0.035175
Email	-0.264068	-0.460741	-0.017828
Contacts	-0.505665	0.268712	0.233107
Yearly_equip_failure	0.324211	0.575115	-0.485935
Tenure	-0.012717	-0.009792	-0.002763
MonthlyCharge	-0.088752	0.026823	0.017526
Bandwidth_GB_Year	0.004384	-0.016344	0.003894
item1	0.001499	0.008175	-0.015603
item2	-0.01154	0.009724	0.000671
item3	-0.004121	-0.033235	-0.026675
item4	-0.005048	-0.006544	-0.019363
item5	-0.02315	0.005373	-0.007718
item6	0.012613	0.009491	0.004988
item7	0.000998	0.004748	0.058498
item8	0.012322	0.033087	-0.009489

Part V: References

G.

The only third-party code I used was the regex to find non-ascii characters. No other third-party code was used.

regex101. (n.d.). Test regex with regex tester. Retrieved March 23, 2023, from https://regex101.com/r/wZo2wB/1

H.

A general reference I used was the DataCamp supplemental material.

DataCamp. (n.d.). Custom D206 data cleaning. Retrieved March 23, 2023, from https://app.datacamp.com/learn/custom-tracks/custom-d206-data-cleaning

No additional sources were used.