D206

Introduction

In the telecommunications industry, customers can choose from multiple service providers and actively switch from one provider to another. Customer "churn" is defined as the percentage of customers who stopped using a provider's product or service during a certain time frame. In this highly competitive market, some telecommunications industries can experience average annual churn rates as high as 25 percent. Given that it costs 10 times more to acquire a new customer than to retain an existing one, customer retention has now become even more important than customer acquisition. For many providers, retaining highly profitable customers is the number one business goal. To reduce customer churn, telecommunications companies need to predict which customers are at high risk of churn. You are an analyst on a team of analysts in a popular telecommunications company, which serves customers in all regions of the United States. You have been asked to clean the raw data set in preparation to explore the data, identify trends, and compare key metrics.

In [1]:
```
# Import Libraries and magic words

import pandas as pd
import numpy as np


import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from helper import *
%matplotlib inline
import random
import warnings

warnings.filterwarnings('ignore')
```

In [2]:
```
# Setting some options
pd.set_option('display.max_columns', None) #this ensures dataframes are not truncated
warnings.filterwarnings("ignore", category = UserWarning, module = "matplotlib") #This notebook shouldn't take
```

In [3]:
```
os.listdir()
```

Out[3]:
```
['D206 Data Cleaning_Churn Data Consideration and Dictionary.pdf',
 'churn_raw_data.csv',
 '.idea',
 '.ipynb_checkpoints',
 '__pycache__',
 'helper.py',
 'd206 Paper.docx',
 'churn_clean.csv',
 'Churn_scaled_data.csv',
 'WGUd206.ipynb',
 'WGUd206.pdf',
 'WGUd206.html']
```

Part I: Research Question

A. Describe one question or decision that you will address using the data set you chose. The summarized question or decision must be relevant to a realistic organizational need or situation.

A dataset was provided from a US telecom provider to begin building insights into their customer base. Understanding the what features contribute most toward knowledge in the dataset is essential in order to build models and make recommendations. This will help preprare for the subsequent reserach question: "Which customers are ,more liekly to churn?".

B. Describe the variables in the data set and indicate the specific type of data being described. Use examples from the data set that support your claims.

The dataset provided is 10,000 observations by 50 features. The dataset came with a data dictionary which is attached. For this analysis, all categorical, binary and numerical data not corresponding to location was used. This means the only features removed are as follows: • CaseOrder (categorical): A placeholder variable to preserve the original order of the raw data file • Customer_id (categorical): Unique customer ID • Interaction (categorical): Unique IDs related to customer transactions, technical support, and sign-ups • City (categorical): Customer city of residence as listed on the billing statement • State (categorical): Customer state of residence as listed on the billing statement • County (categorical): Customer county of residence as listed on the billing statement • Zip (categorical): Customer zip code of residence as listed on the billing statement • Lat, Lng(categorical): GPS coordinates of customer residence as listed on the billing statement • Area(categorical): Area type (rural, urban, suburban), based on census data

The following items in the dataset are categorical features included in the dataset • TimeZone (categorical): Time zone of customer residence based on customer's sign-up information • Job (categorical): Job of the customer (or invoiced person) as reported in sign-up information • Education (categorical): Highest degree earned by customer as reported in sign-up information • Employment (categorical): Employment status of customer as reported in sign-up information • Marital (categorical): Marital status of customer as reported in sign-up information • Gender (categorical): Customer self-identification as male, female, or nonbinary • Churn (categorical): Whether the customer discontinued service within the last month (yes, no) • Techie (categorical): Whether the customer considers themselves technically inclined (based on customer questionnaire when they signed up for services) (yes, no) • Contract (categorical): The contract term of the customer (month-to-month, one year, two year) • Port_modem (categorical): Whether the customer has a portable modem (yes, no) • Tablet (categorical): Whether the customer owns a tablet such as iPad, Surface, etc. (yes, no) • InternetService (categorical): Customer's internet service provider (DSL, fiber optic, None) • Phone (categorical): Whether the customer has a phone service (yes, no) • Multiple (categorical): Whether the customer has multiple lines (yes, no) • OnlineSecurity (categorical): Whether the customer has an online security add-on (yes, no) • OnlineBackup (categorical): Whether the customer has an online backup add-on (yes, no) • DeviceProtection (categorical): Whether the customer has device protection add-on (yes, no) • TechSupport (categorical): Whether the customer has a technical support add-on (yes, no) • StreamingTV (categorical): Whether the customer has streaming TV (yes, no) • StreamingMovies (categorical): Whether the customer has streaming movies (yes, no) • PaperlessBilling (categorical): Whether the customer has paperless billing (yes, no) • PaymentMethod (categorical): The customer's payment method (electronic check, mailed check, bank (automatic bank transfer), credit card (automatic))

The following variables represent response to an eight-question survey asking customers to rate the importance of various factors/surfaces on a scale of 1 to 8 (1 = most important, 8 = least important) • Item1 (categorical): Timely response • Item2 (categorical): Timely fixes • Item3 (categorical): Timely replacements • Item4 (categorical): Reliability • Item5 (categorical): Options • Item6 (categorical): Respectful response • Item7 (categorical): Courteous exchange • Item8 (categorical): Evidence of active listening

The following features were used to weigh principal components in the analysis: • Population (numeric): Population within a mile radius of customer, based on census data • Children (numeric): Number of children in customer's household as reported in sign-up information • Age (numeric): Age of customer as reported in sign-up information • Income (numeric): Annual income of customer as reported at time of sign-up • Outage_sec_perweek (numeric): Average number of seconds per week of system outages in the customer's neighborhood • Email (numeric): Number of emails sent to the customer in the last year (marketing or correspondence) • Contacts (numeric): Number of times customer contacted technical support • Yearly_equip_failure (numeric): The number of times customer's equipment failed and had to be reset/replaced in the past year • Tenure (numeric): Number of months the customer has stayed with the provider • MonthlyCharge (numeric): The amount charged to the customer monthly. This value reflects an average per customer. • Bandwidth_GB_Year (numeric): The average amount of data used, in GB, in a year by the customer

Part II: Data-Cleaning Plan

C. Explain the plan for cleaning the data by doing the following:

1. Propose a plan that includes the relevant techniques and specific steps needed to identify anomalies in the data set.

Phase 1 - Using common Python libraries for descriptive statistics and visualization, data will be assessed for: a. Appropriate typing: using Pandas info instantiation will show feature typing. b. Missing Values; using Pandas Info instantiation, an overview of the original file's columns will be displayed with how many observations each feature has. c. Duplicate Values: using the Pandas duplicated class, a sum of identical rows will be documented. d. Numerical Distribution: using matplotlib histograms alongside box and whisker plots, unclean data will be visualized. e. Categorical accuracy; assessment of categorical column inputs will be viewed and documented if a recommended update can be made. Also, noting what items should be encoded via Scikit-learn 1-hot Encoding (binary and multivariate) if machine learning is to be used in the future. f. Ensure input values make sense and correct outliers. For example, there should be no negative numbers in this dataset based on the descriptions in the data dictionaries

These are standard data assessment procedures (Yıldırım, 2020).

Phase 2 – Cleaning data with Pandas by:

a. Imputing missing values or negative values is dependent on the type of data present and its distribution (Kumar, 2021).

b. Missing values imputation i. Numerical data will be cleaned depending on its distribution. Distributions will be be explored with descriptive stats functions and visualizations ii. Normal or even distribution will impute randomly from the sample as a standard practice. (Kumar, 2021) iii. Bimodal data will impute the median of each mode splitting the distribution in half. This works better than random sample as it minimizes risk and variance. (IBM, 2021) iv. Negative values on time data will be set to 0 in numerical datatypes. v. Note that K-Nearest-Neighbors may fill this dataset accurately as opposed to manual techniques, I chose not to do this due to the high dimensionality and large scale difference between

c. Duplicate rows will be dropped

d. Scaling data via ScikitLearn RobustScaler

e. Categorical accuracy; assessment of categorical column inputs will be viewed and documented if a recommended update can be made. Also, noting what items should be encoded via Scikit-learn 1-hot Encoding (binary and multivariate).

2. Justify your approach for assessing the quality of the data, include:

The characteristics of the data being assessed:

The dataset has mixed features, often self reported by the customer for categories and some discrete data. For example, income is self reported for the data and therefore correcting for outliers is not possible without another dataset to compare against. Also, categories such as `Techie` and `TechSupport` having missing values speaks to data collection process having major errors from the firm. Every dataset should have direct knowledge on whether a customer has purchased a service or not.

The approach used to assess the quality:

Using graphic visualization via Seaborn as well as descriptive stats via standard Pandas tools is the standard approach for initial exploration.

3. Justify your selected programming language and any libraries and packages that will support the data-cleaning process.

Python is the most used programming language for analysis computation as of writing this. With its robust package selection, statistics, machine learning and ease of use make it an ideal candidate for this type of analysis work. The JupyterNotebook is a data science and analytics industry standard for human-readable code and experiment reproducibility. Pandas allows working with Dataframes and is the foundational method I use to explore this dataset.

4. Provide the code you will use to identify the anomalies in the data.

Initial Exploration

This section servers as a playground using different exploratory options to help identify data issues. Not every issue found will have a cell but there will be a documented section below some of the more overarching and larger calls will be displayed below.

In [4]:
```
#Read in Dataset
raw_churn = pd.read_csv('churn_raw_data.csv')
```

In [5]:
```
# Dimensions
raw_churn.shape
```

Out[5]:
```
(10000, 52)
```

In [6]:
```
# Defualt ordering column
raw_churn.index
```

Out[6]:
```
RangeIndex(start=0, stop=10000, step=1)
```

In [7]:
```
# Pandas dataset descriptor
raw_churn.info() # Note data type anomalies
```

```
<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
```

1. Unused empty column
2. `CaseOrder` should be index
3. `Zip` / `Lat` / `Lang` should be string/panda objects
4. Columns missing data: `Children`, `Age`, `Income`, `Techie`, `Phone`, `TechSupport`, `Tenure`, `Bandwidth-GB-Year`
5. Survey `item` header is non descript

In [8]:
```
# Descriptive stats to detect data anomalies in numeric columns
with pd.option_context('display.max_columns', 40):
    print(raw_churn.describe(include = [np.number]))
```

```
        Unnamed: 0    CaseOrder           Zip           Lat           Lng  \
count  10000.00000  10000.00000  10000.000000  10000.000000  10000.000000
mean    5000.50000   5000.50000  49153.319600     38.757567    -90.782536
std     2886.89568   2886.89568  27532.196108      5.437389     15.156142
min        1.00000      1.00000    601.000000     17.966120   -171.688150
25%     2500.75000   2500.75000  26292.500000     35.341828    -97.082812
50%     5000.50000   5000.50000  48869.500000     39.395800    -87.918800
75%     7500.25000   7500.25000  71866.500000     42.106908    -80.088745
max    10000.00000  10000.00000  99929.000000     70.640660    -65.667850

          Population     Children          Age         Income  \
count   10000.000000  7505.000000  7525.000000    7510.000000
mean     9756.562400     2.095936    53.275748   39936.762226
std     14432.698671     2.154758    20.753928   28358.469482
min         0.000000     0.000000    18.000000     740.660000
25%       738.000000     0.000000    35.000000   19285.522500
50%      2910.500000     1.000000    53.000000   33186.785000
75%     13168.000000     3.000000    71.000000   53472.395000
max    111850.000000    10.000000    89.000000  258900.700000

       Outage_sec_perweek         Email      Contacts  Yearly_equip_failure  \
count        10000.000000  10000.000000  10000.000000          10000.000000
mean            11.452955     12.016000      0.994200              0.398000
std              7.025921      3.025898      0.988466              0.635953
min             -1.348571      1.000000      0.000000              0.000000
25%              8.054362     10.000000      0.000000              0.000000
50%             10.202896     12.000000      1.000000              0.000000
75%             12.487644     14.000000      2.000000              1.000000
max             47.049280     23.000000      7.000000              6.000000

            Tenure  MonthlyCharge  Bandwidth_GB_Year         item1  \
count  9069.000000   10000.000000        8979.000000  10000.000000
mean     34.498858     174.076305        3398.842752      3.490800
std      26.438904      43.335473        2187.396807      1.037797
min       1.000259      77.505230         155.506715      1.000000
25%       7.890442     141.071078        1234.110529      3.000000
50%      36.196030     169.915400        3382.424000      3.000000
75%      61.426670     203.777441        5587.096500      4.000000
max      71.999280     315.878600        7158.982000      7.000000

              item2         item3         item4         item5         item6  \
count  10000.000000  10000.000000  10000.000000  10000.000000  10000.000000
mean       3.505100      3.487000      3.497500      3.492900      3.497300
std        1.034641      1.027977      1.025816      1.024819      1.033586
min        1.000000      1.000000      1.000000      1.000000      1.000000
25%        3.000000      3.000000      3.000000      3.000000      3.000000
50%        4.000000      3.000000      3.000000      3.000000      3.000000
75%        4.000000      4.000000      4.000000      4.000000      4.000000
max        7.000000      8.000000      7.000000      7.000000      8.000000

              item7         item8
count  10000.000000  10000.000000
mean       3.509500      3.495600
std        1.028502      1.028633
min        1.000000      1.000000
25%        3.000000      3.000000
50%        4.000000      3.000000
75%        4.000000      4.000000
max        7.000000      8.000000
```

1. `Population` has an area with 0 people.
2. `Income` has a min of 740, due to this being self reported and no comparison this analysis cannot accurately asses the validity of and income oddities
3. `Outage_sec_perweek` has negative numbers, this does not make sense for time data.

In [9]:
```
#check for duplicates
if raw_churn.duplicated().sum() == 0:
    print('No Duplicates')
```

```
No Duplicates
```

In [10]:
```
# Preview Dataset
raw_churn.head()
```

Out[10]:

	Unnamed: 0	CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	Area	
0	1	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	38	Urban	Am
1	2	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	10446	Urban	Amer
2	3	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	Urban	America/Lo
3	4	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32.96687	-117.24798	13863	Suburban	America/Lo
4	5	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	Suburban	Americ

In [11]:
```
#seperate numeric datatypes and non numeric

numeric_col_list = []
nonnumeric_col_list = []
for i in raw_churn:
    try:
        if raw_churn[i][0] > -100000000000:
            numeric_col_list.append(i)
    except Exception:
        nonnumeric_col_list.append(i)
        pass
```

In [12]:
```
# view lists created for analysis
numeric_col_list, nonnumeric_col_list
```

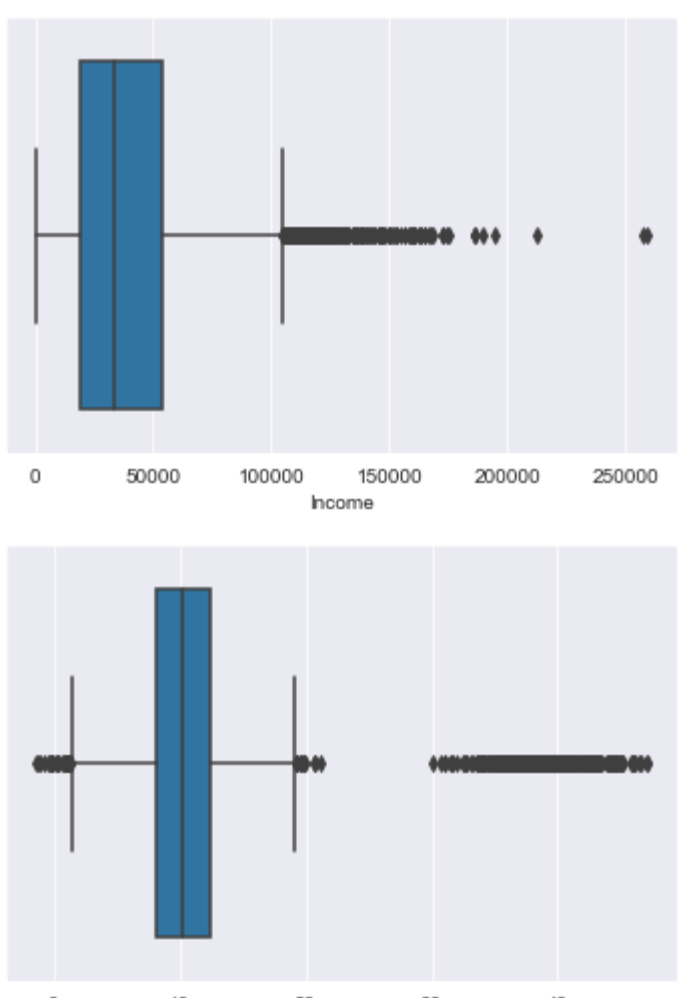
Out[12]:
```
(['Unnamed: 0',
  'CaseOrder',
  'Zip',
  'Lat',
  'Lng',
  'Population',
  'Age',
  'Income',
  'Outage_sec_perweek',
  'Email',
  'Contacts',
  'Yearly_equip_failure',
  'Tenure',
  'MonthlyCharge',
  'Bandwidth_GB_Year',
  'item1',
  'item2',
  'item3',
  'item4',
  'item5',
  'item6',
  'item7',
  'item8'],
 ['Customer_id',
  'Interaction',
  'City',
  'State',
  'County',
  'Area',
  'Timezone',
  'Job',
  'Education',
  'Employment',
  'Marital',
  'Gender',
  'Churn',
  'Techie',
  'Contract',
  'Port_modem',
  'Tablet',
  'InternetService',
  'Phone',
  'Multiple',
  'OnlineSecurity',
  'OnlineBackup',
  'DeviceProtection',
  'TechSupport',
  'StreamingTV',
  'StreamingMovies',
  'PaperlessBilling',
  'PaymentMethod'])
```

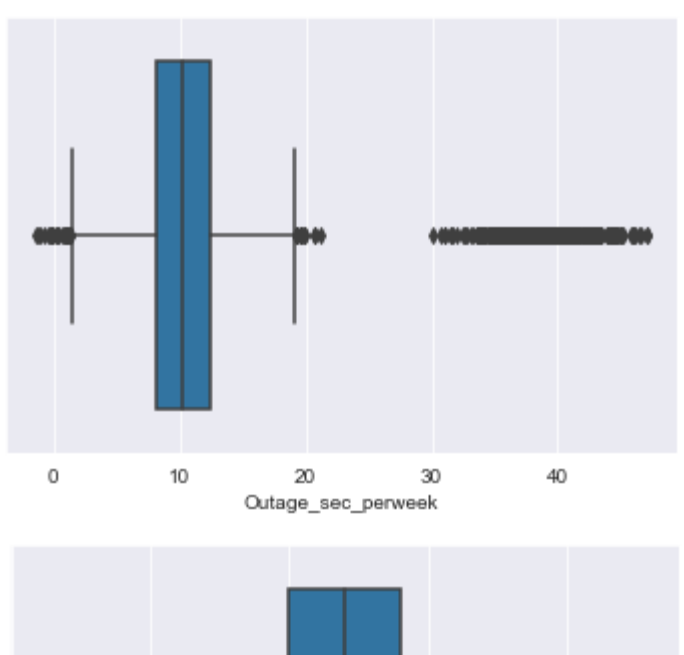
In [13]:
```
#some preliminary cleaning, pop labels that will be unused due to wrong typing or uniqueness
to_pop = ['Unnamed: 0',
          'CaseOrder',
          'Zip',
          'Lat',
          'Lng']

for popper in to_pop:
    try:
        numeric_col_list.remove(popper)
    except Exception:
        pass
```

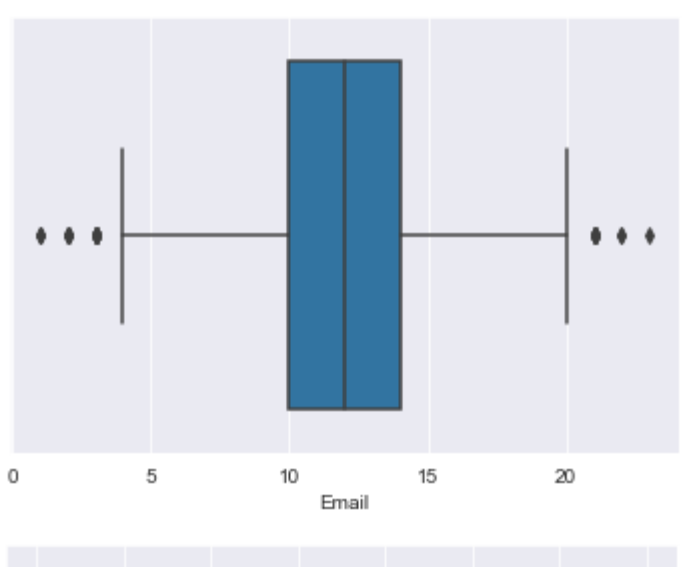
In [14]:
```
#unit test
numeric_col_list, nonnumeric_col_list
```

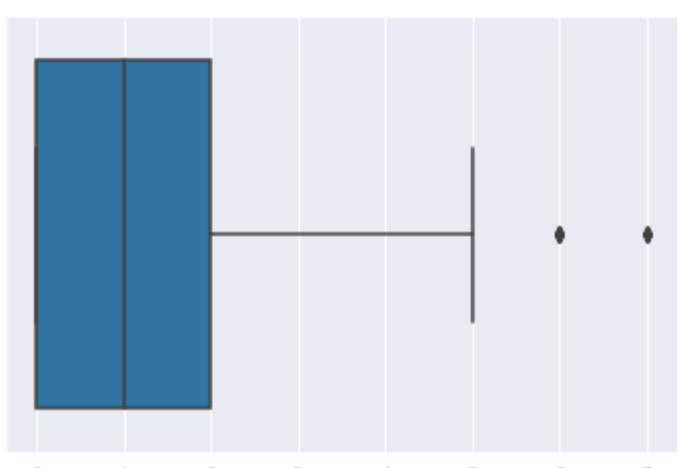
Out[14]:
```
(['Population',
  'Age',
  'Income',
  'Outage_sec_perweek',
  'Email',
  'Contacts',
  'Yearly_equip_failure',
  'Tenure',
  'MonthlyCharge',
  'Bandwidth_GB_Year',
  'item1',
  'item2',
  'item3',
  'item4',
  'item5',
  'item6',
  'item7',
  'item8'],
 ['Customer_id',
  'Interaction',
  'City',
  'State',
  'County',
  'Area',
  'Timezone',
  'Job',
  'Education',
  'Employment',
  'Marital',
  'Gender',
  'Churn',
  'Techie',
  'Contract',
  'Port_modem',
  'Tablet',
  'InternetService',
  'Phone',
  'Multiple',
  'OnlineSecurity',
  'OnlineBackup',
  'DeviceProtection',
  'TechSupport',
  'StreamingTV',
  'StreamingMovies',
  'PaperlessBilling',
  'PaymentMethod'])
```

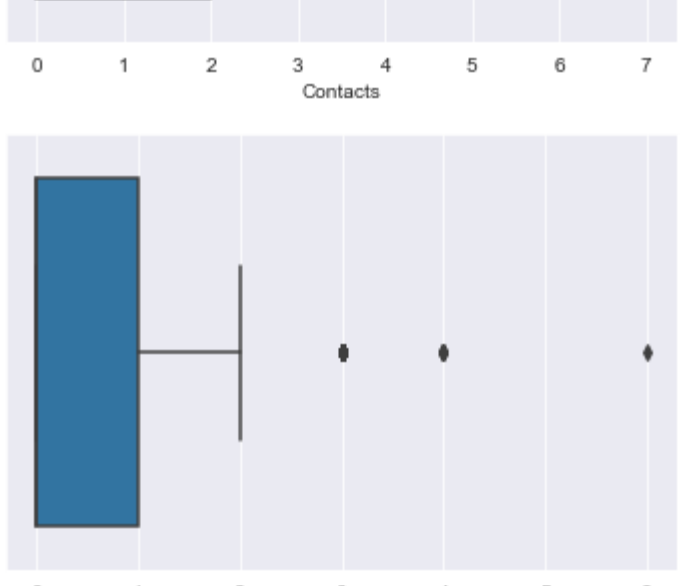
In [15]:
```
#visualize numeric distributions boxbplot
for i, col in enumerate(raw_churn[numeric_col_list]):
        try:
            plt.figure(i)
            sns.boxplot(raw_churn[col])
        except Exception:
            pass
```

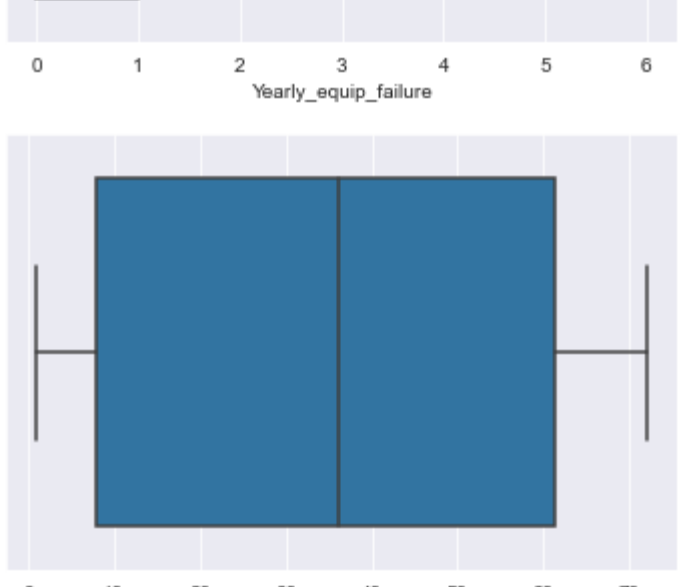


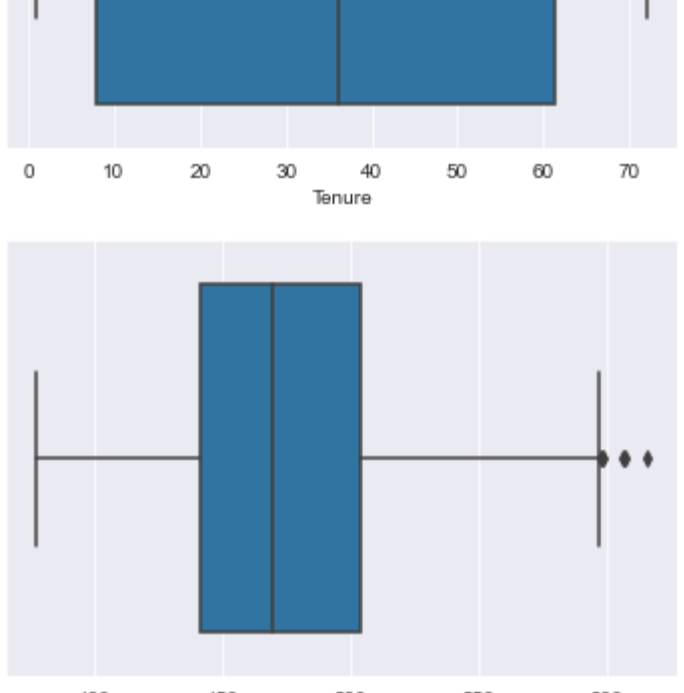


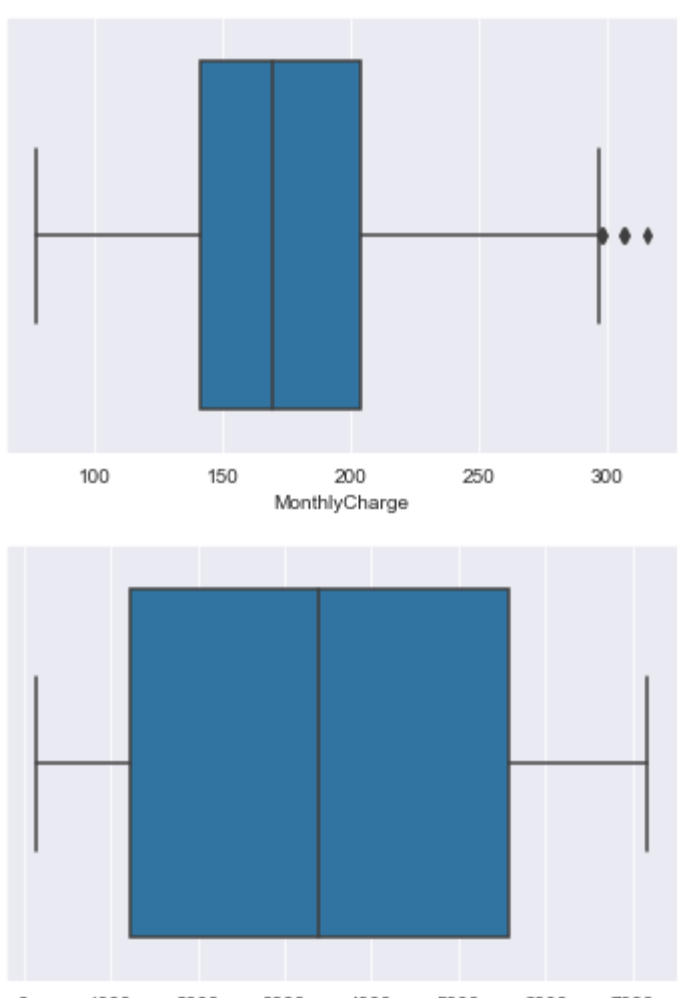


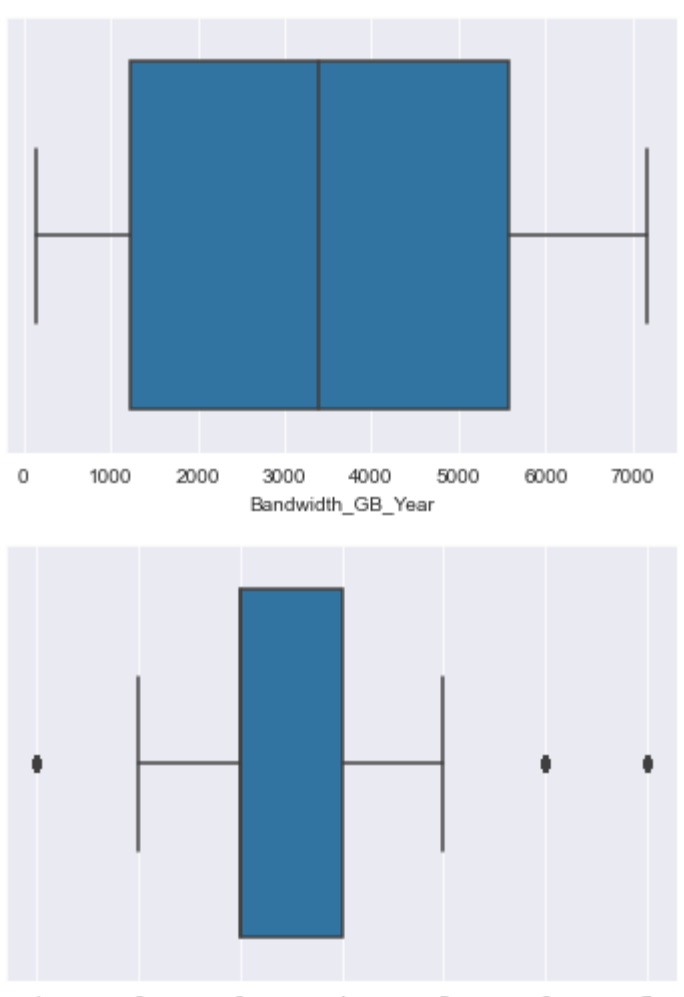


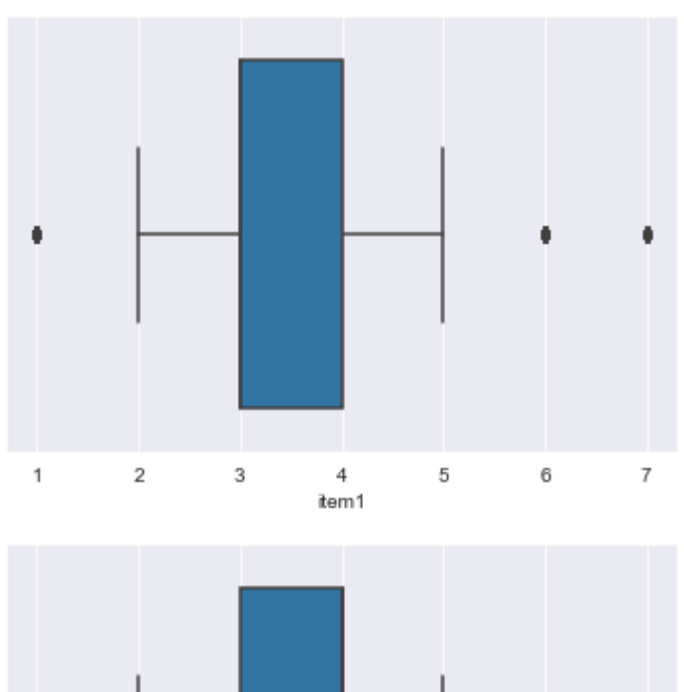


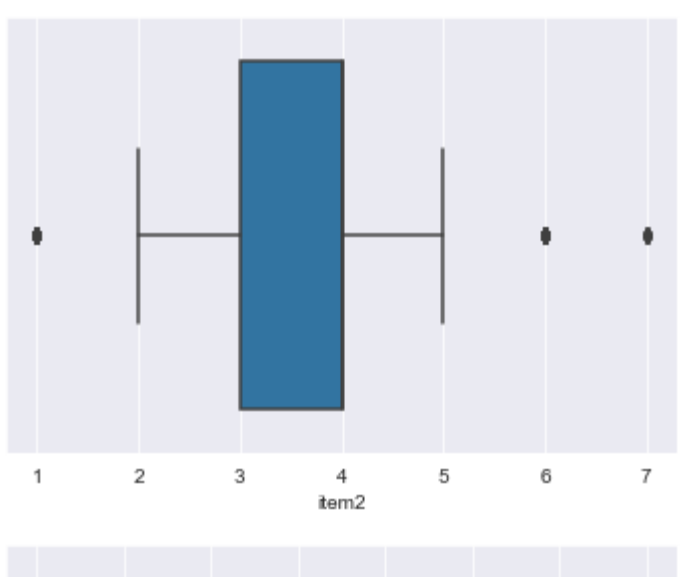


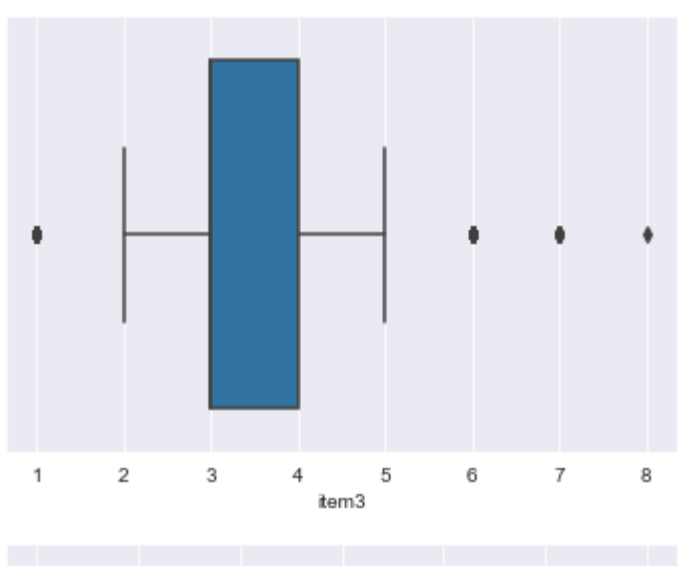


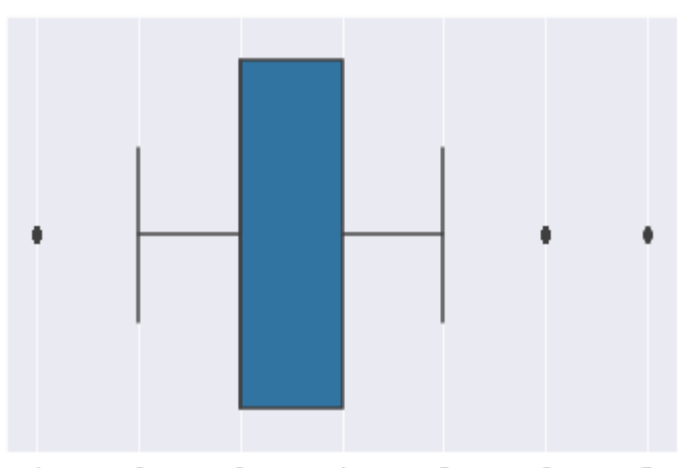


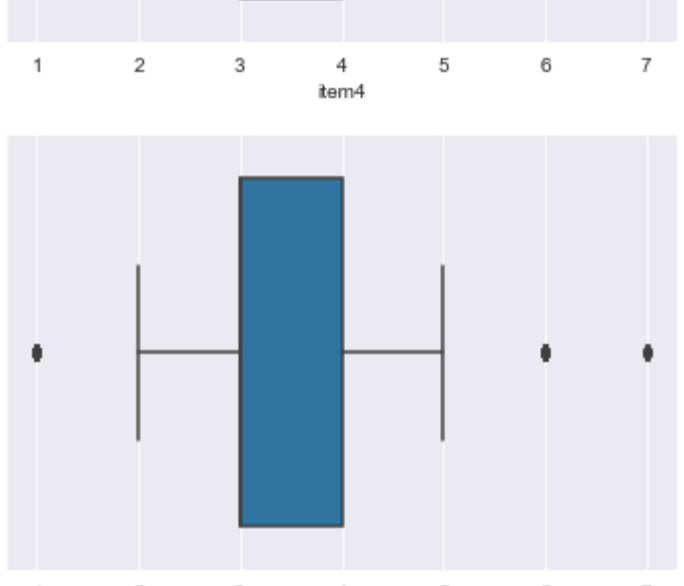


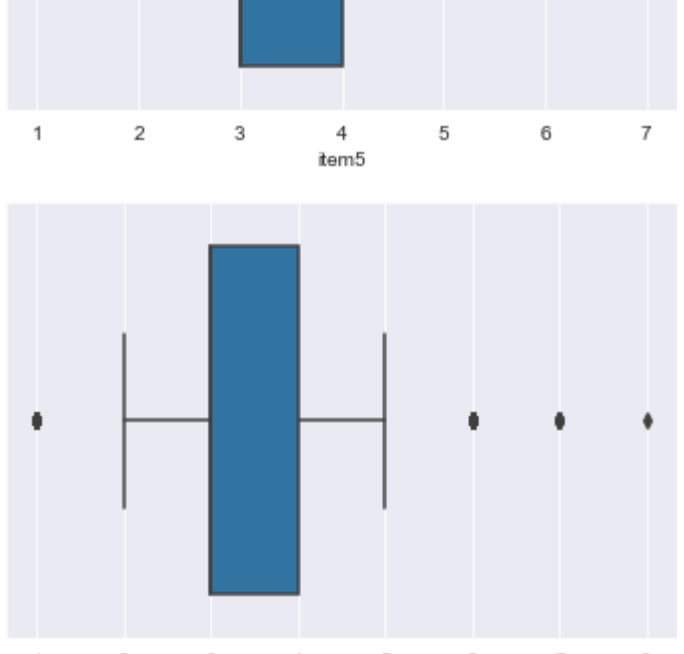


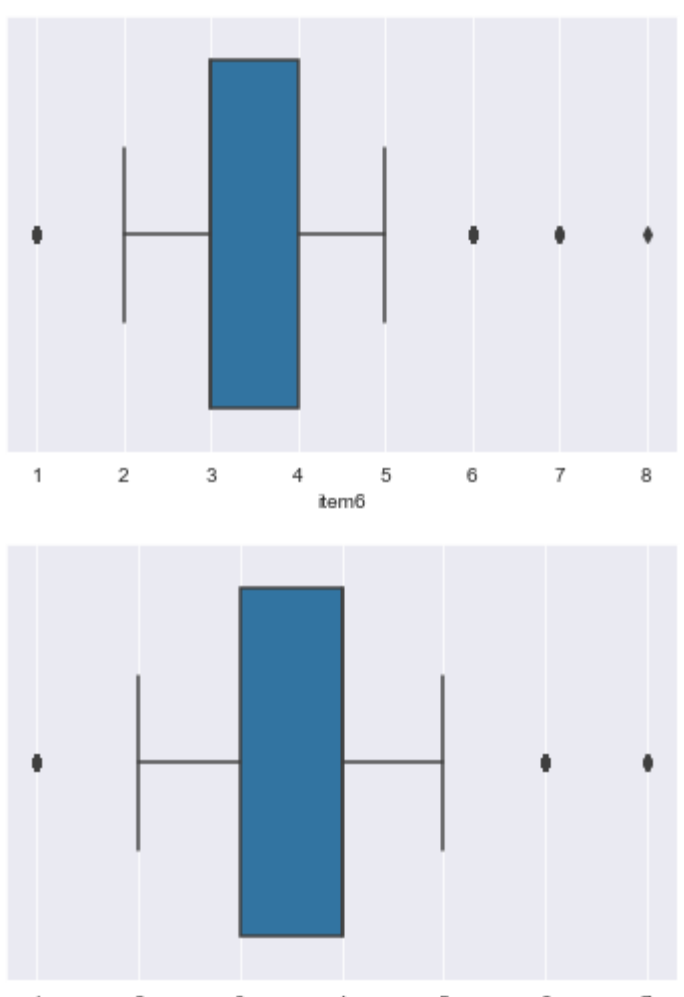


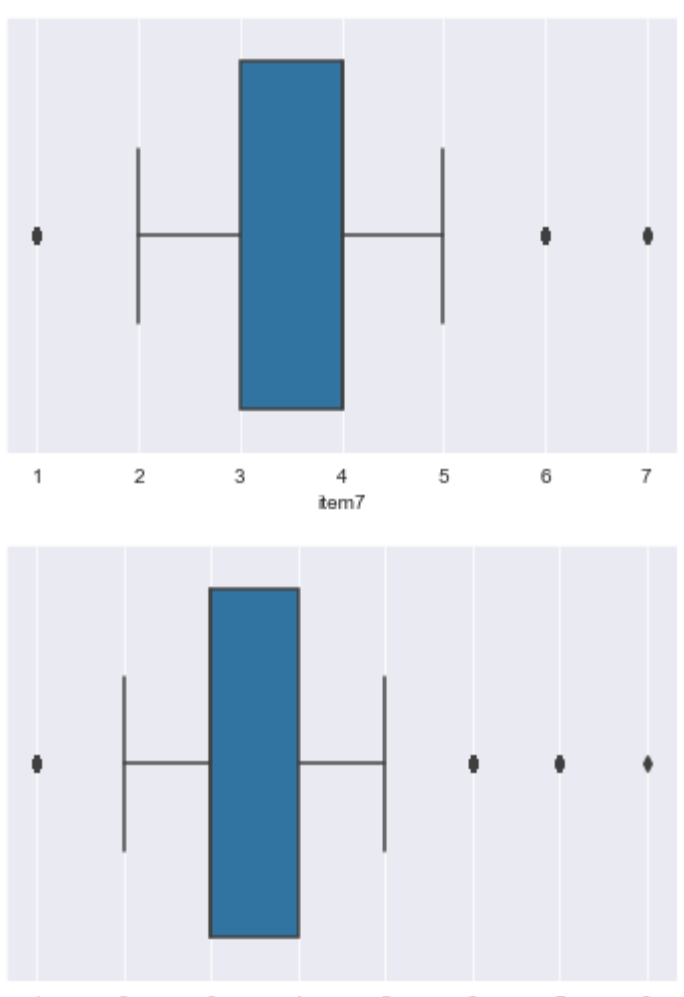


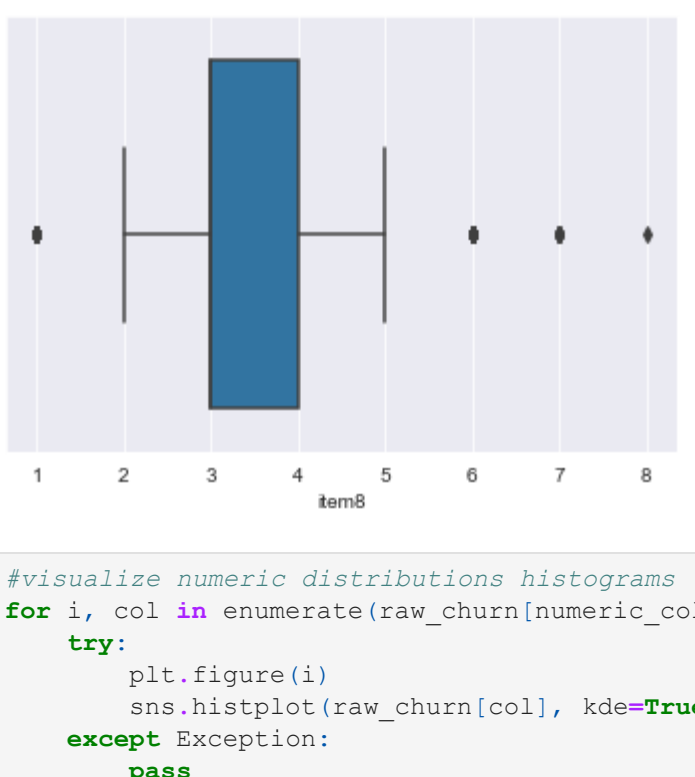












In [16]:
```
#visualize numeric distributions histogram
for i, col in enumerate(raw_churn[numeric_col_list]):
    try:
        plt.figure(i)
        sns.histplot(raw_churn[col], kde=True)
    except Exception:
        pass
```

In [46]:
```
#fill na

for col in cat_nulls:
    df_clean[col].fillna('No', inplace=True)
    #built in unit test
    print(df_clean[col].value_counts(), len(df_clean[col]))
```
```
No     8743
Yes    1257
Name: Techie, dtype: int64 10000
Yes    8128
No     1872
Name: Phone, dtype: int64 10000
No     6626
Yes    3374
Name: TechSupport, dtype: int64 10000
```

Issue 4

Empty unnamed first column that corresponds to index should be dropped and Case Order should be set to index

Using drop and set index, a proper index well be made. This de-dupes data and structures it better.

In [47]:
```
# View Unnamed column
df_clean.iloc[:, 0:1].head()
```
Out[47]:

	Unnamed: 0
0	1
1	2
2	3
3	4
4	5

In [48]:
```
# Drop column inplace
df_clean.drop(df_clean.columns[0],axis=1,inplace=True)
```
In [49]:
```
# drop and reset index
df_clean = df_clean.reset_index(drop=True)
df_clean.set_index('CaseOrder', inplace=True)
```
In [50]:
```
#unit test verification
df_clean.head()
```
Out[50]:

CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	Area	Timezone
1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.251	-133.37571	38	Urban	America/Sitka
2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44.32893	-84.2408	10446	Urban	America/Detroit
3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	Urban	America/Los_Angeles
4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32.96687	-117.24798	13863	Suburban	America/Los_Angeles
5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	Suburban	America/Chicago

Issue 5

Education levels should be standardized. Using a dictionary created here, I am standardizing the customer's education levels based on US Census standards. The current list is unwieldy and too granular to make general insights.

In [51]:
```
df_clean['Education'].value_counts()
```
Out[51]:
```
Regular High School Diploma                 2421
Bachelor's Degree                           1703
Some College, 1 or More Years, No Degree    1562
9th Grade to 12th Grade, No Diploma          870
Master's Degree                              764
Associate's Degree                           760
Some College, Less than 1 Year               652
Nursery School to 8th Grade                  449
GED or Alternative Credential                387
Professional School Degree                   198
No Schooling Completed                       118
Doctorate Degree                             116
Name: Education, dtype: int64
```
In [52]:
```
education_levels ={"Master's Degree": 'Graduate Degree',
                   'Regular High School Diploma': 'Highschool',
                   'Doctorate Degree': 'Graduate Degree',
                   'No Schooling Completed': 'No Diploma',
                   "Associate's Degree": 'Associate\'s/Some College',
                   "Bachelor's Degree": 'Bachelor\'s Degree',
                   'Some College, Less than 1 Year': 'Associate\'s/Some College',
                   'GED or Alternative Credential': 'Graduate Degree',
                   'Some College, 1 or More Years, No Degree': 'Associate\'s/Some College',
                   '9th Grade to 12th Grade, No Diploma': 'No Diploma',
                   'Nursery School to 8th Grade': 'No Diploma',
                   'Professional School Degree':'Graduate Degree'}

# use mapping to the above dictionary

df_clean['Education'] = df_clean['Education'].map(education_levels)
```
In [53]:
```
# Unit Test

df_clean['Education'].value_counts()
```
Out[53]:
```
Associate's/Some College    2974
Highschool                  2421
Bachelor's Degree           1703
Graduate Degree             1465
No Diploma                  1437
Name: Education, dtype: int64
```

Issue 6

Rename Survey `item` columns to be more descriptive. This allows easier parsing of data meaning

In [54]:
```
df_clean.rename(columns = {'item1':'Timeliness',
                     'item2':'Fixes',
                     'item3':'Replacements',
                     'item4':'Reliability',
                     'item5':'Options',
                     'item6':'Respectfulness',
                     'item7':'Courteous',
                     'item8':'Listening'},
          inplace=True)
```
In [55]:
```
#unit test
df_clean.columns
```
Out[55]:
```
Index(['Customer_id', 'Interaction', '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', 'Timeliness', 'Fixes', 'Replacements',
       'Reliability', 'Options', 'Respectfulness', 'Courteous', 'Listening'],
      dtype='object')
```

D5 Provide a copy of the cleaned data set.

The data has been cleaned and structured. A CSV with the cleaned structure will be saved.

In [56]:
```
df_clean.to_csv('churn_clean.csv')
```

D6 Summarize the limitations of the data-cleaning process.

." The limitations of this dataset can be traced back to the data source. The data was voluntarily reported and therefore quality and accuracy can only be verified from more strenuous income matching via credit checks or employment verification. My imputation techniques also may be less effective than K-Nearest-Neighbors or decision tree classifiers such as random forest or a decision tree. Those methods however are more computationally expensive and may be impacted further by the quality of self-reported data. This seriously impacts the ability ot do meaningful hypothesis testing or ANOVA with the current cleaned data.

Some Quality issues include age pairing with employment and children. The customer 18 year of age labeled as retired with and with multiple kids. Some of these younger people are labeled as widowed or divorced as well. There is not a reasonable way to adjust these entries due to being self reported. K-Nearest Neighbors would also cause the dataset to impute incorrectly with these quality issues.

Other anomalies come from data the firm should have. A telecom provider giving a customer dataset should not have null values for its services per customer. A re-evalution of data collection techniques should be used. It is hard to know if the imputed data is consistent with real world data.

D7 Discuss how the limitations in part D6 affect the analysis of the question or decision from part A.

These limitations point to using this dataset only for tentative testing. For example, Age had 25% of its data missing. With that missing data assuming the distribution was valid, bias can be introduced into the dataset that would not be reflective of the real customer ages. This could also impact machine learning training in ways causing bad predictions.

PCA

Approach

PCA will be applied to the continuous variables in the dataset. Before applying PCA, the data will be scaled via RobustScaler. Many of the items in this dataset were self-reported causing anomalies in data quality and mismatched categorization. For example, a customer in Yamhill Oregon is noted to be a CFO but income is listed $33000. Due to these outliers and the nature of this data being a self-reported survey, providing treatment corrections to this data without another similar sample is not recommended. Instead, RobustScaler limits the effects of outliers by assuming the interquartile range of data correctly corresponds to the data distribution (Hale, 2019)

The scaled data will consist of the following continuous variables: ['Outage_sec_perweek', 'Yearly_equip_failure', 'Age', 'Contacts', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year']

In [57]:
```
# set scaler
from sklearn.preprocessing import RobustScaler, StandardScaler

# instantiate scaler
scaler = RobustScaler()
```
In [58]:
```
# set scaling metrics
cont_cols = ['Outage_sec_perweek', 'Yearly_equip_failure', 'Age', 'Contacts', 'Tenure', 'MonthlyCharge', 'Bandw
```
In [59]:
```
# a copy of the clean dataframe will be made to preserve dataframe integrity

df_scaled = df_clean.copy()
```
In [60]:
```
# test copy
df_scaled.head()
```
Out[60]:

CaseOrder	Customer_id	Interaction	City	State	County	Zip	Lat	Lng	Population	Area	Timezone
1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56.251	-133.37571	38	Urban	America/Sitka
2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44.32893	-84.2408	10446	Urban	America/Detroit
3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	Urban	America/Los_Angeles
4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32.96687	-117.24798	13863	Suburban	America/Los_Angeles
5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	Suburban	America/Chicago

In [61]:
```
# preview continuous variables descriptive stats as a baseline to review after scaling
df_scaled[cont_cols].describe()
```
Out[61]:

	Outage_sec_perweek	Yearly_equip_failure	Age	Contacts	Tenure	MonthlyCharge	Bandwidth_GB_Year
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	11.453592	0.398000	53.284200	0.994200	36.966114	174.076305	3177.812307
std	7.024842	0.635953	20.747402	0.988466	26.329346	43.335473	2173.895637
min	0.000000	0.000000	18.000000	0.000000	1.000259	77.505230	155.506715
25%	8.054362	0.000000	35.000000	0.000000	8.700329	141.071078	1234.000000
50%	10.202896	0.000000	53.000000	1.000000	47.444800	169.915400	2046.732140
75%	12.487644	1.000000	71.000000	2.000000	61.000000	203.777441	5466.284500
max	47.049280	6.000000	89.000000	7.000000	71.999280	315.878600	7158.982000

In [62]:
```
# apply scaler
df_scaled[cont_cols] = scaler.fit_transform(df_scaled[cont_cols])
```
In [63]:
```
#unit test scaling
df_scaled[cont_cols].describe()
```
Out[63]:

	Outage_sec_perweek	Yearly_equip_failure	Age	Contacts	Tenure	MonthlyCharge	Bandwidth_GB_Year
count	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	1.000000e+04
mean	2.821151e-01	0.398000	0.007894	-0.002900	-0.200359	0.066355	2.672505e-01
std	1.584569e+00	0.635953	0.576317	0.494233	0.503432	0.691086	5.136459e-01
min	-2.301432e+00	0.000000	-0.972222	-0.500000	-0.888047	-1.473697	-4.468569e-01
25%	-4.846374e-01	0.000000	-0.500000	-0.500000	-0.740817	-0.459990	-1.920315e-01
50%	-2.003606e-16	0.000000	0.000000	0.000000	0.000000	0.000000	2.688821e-17
75%	5.153626e-01	1.000000	0.500000	0.500000	0.259183	0.540010	8.079685e-01
max	8.311311e+00	6.000000	1.000000	3.000000	0.469496	2.327726	1.207917e+00

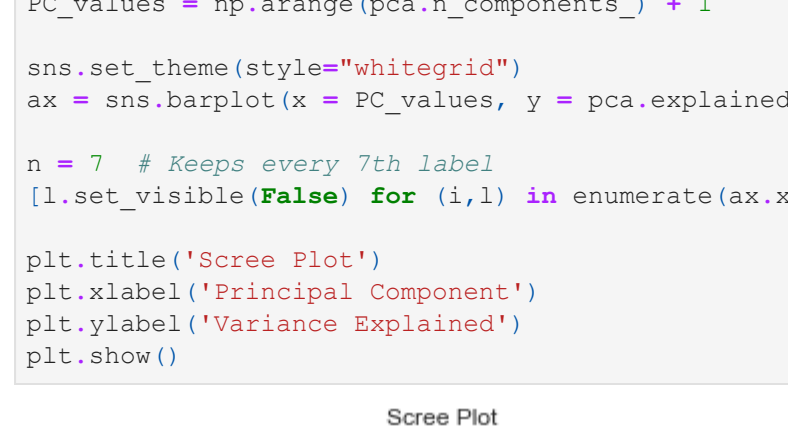
In [64]:
```
# save the scaled data to csv for future use
df_scaled.to_csv('Churn_scaled_data.csv')
```

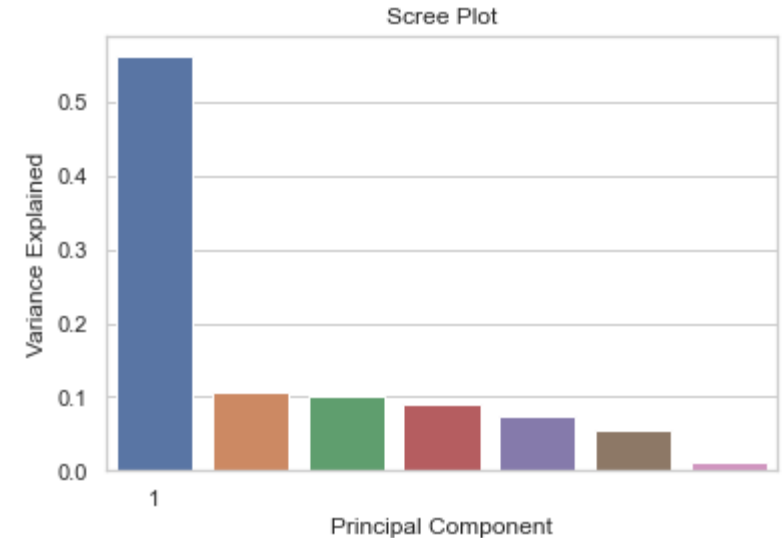
PCA

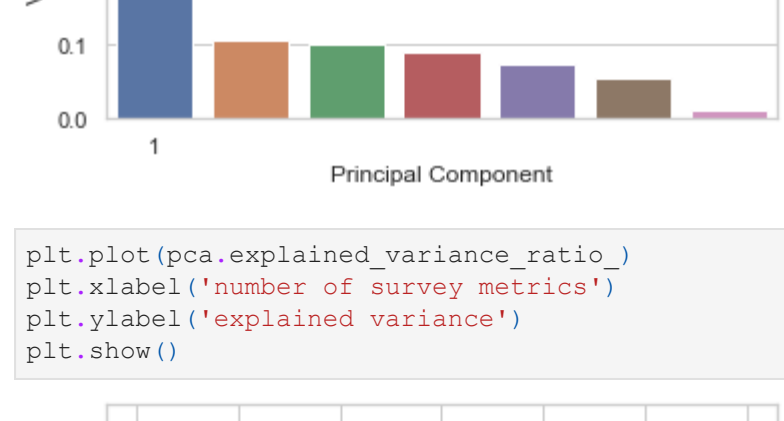
Now PCA will be applied to the continuous data

In [65]:
```
from sklearn.decomposition import PCA

#create a dataframe with the scaled continuous variables

df_pca = df_scaled[cont_cols]
df_pca.reset_index(drop=True, inplace=True)
df_pca.head()
```
Out[65]:

	Outage_sec_perweek	Yearly_equip_failure	Age	Contacts	Tenure	MonthlyCharge	Bandwidth_GB_Year
0	-0.728654	1.0	0.416667	-0.5	-0.777238	0.024469	-0.269877
1	0.408646	1.0	-0.722222	-0.5	-0.885056	1.164676	-0.294344
2	0.009636	1.0	-0.083333	-0.5	-0.605944	-0.167048	0.001884
3	1.128576	0.0	-0.138889	0.5	-0.580454	-0.792039	0.027845
4	-0.280284	1.0	0.833333	0.5	-0.875222	-0.305458	-0.419452

In [66]:
```
pca = PCA(7)
X_pca = pca.fit_transform(df_pca)
```
In [67]:
```
PC_values = np.arange(pca.n_components_) + 1

vals = pca.explained_variance_ratio_
cumvals = np.cumsum(vals)
ax = plt.subplot(111)

plt.bar(PC_values, pca.explained_variance_ratio_, color='blue')
ax.plot(PC_values, cumvals)
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```


In [68]:
```
PC_values = np.arange(pca.n_components_) + 1

sns.set_theme(style="whitegrid")
ax = sns.barplot(x = PC_values, y = pca.explained_variance_ratio_)

n = 7  # Keeps every 7th label
[l.set_visible(False) for (i,l) in enumerate(ax.xaxis.get_ticklabels()) if i % n != 0]

plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```


In [69]:
```
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('number of survey metrics')
plt.ylabel('explained variance')
plt.show()
```


In [70]:
```
pca.explained_variance_ratio_[0]
```
Out[70]:
```
0.5619298488812782
```
In [71]:
```
# function to find PCA weights

def weight(x,n,df):
    pca_weights = pd.DataFrame(x.components_, columns=(df.columns))
    weights = pca_weights.iloc[n].sort_values(ascending=False)
    return weights
```
In [72]:
```
len(df_pca.columns)
```
Out[72]:
```
7
```

E1. List the principal components in the data set.

In [73]:
```
for i in range(len(df_pca.columns)):
    print(f'Component {i} explains {int(round(pca.explained_variance_ratio_[i],2) * 100)}% of the data with the
```
```
Component 0 explains 56% of the data with these features contributing the most info:
  Outage_sec_perweek      0.997492
MonthlyCharge           0.069896
Yearly_equip_failure    0.008047
Bandwidth_GB_Year       0.006229
Tenure                  0.003297
Contacts               -0.000583
Age                    -0.002932
Name: 0, dtype: float64

Component 1 explains 11% of the data with these features contributing the most info:
  MonthlyCharge           0.665526
Bandwidth_GB_Year       0.548616
Tenure                  0.503229
Yearly_equip_failure    0.010846
Contacts                0.006533
Age                     0.003618
Outage_sec_perweek     -0.051797
Name: 1, dtype: float64

Component 2 explains 10% of the data with these features contributing the most info:
  MonthlyCharge           0.739527
Age                     0.027677
Contacts                0.001817
Outage_sec_perweek     -0.046560
Yearly_equip_failure   -0.090213
Bandwidth_GB_Year      -0.457025
Tenure                 -0.482858
Name: 2, dtype: float64

Component 3 explains 9% of the data with these features contributing the most info:
  Yearly_equip_failure    0.995443
MonthlyCharge           0.058535
Age                     0.027901
Contacts               -0.011012
Outage_sec_perweek     -0.011607
Bandwidth_GB_Year      -0.044996
Tenure                 -0.051066
Name: 3, dtype: float64

Component 4 explains 7% of the data with these features contributing the most info:
  Age                     0.997781
Contacts                0.051258
Tenure                  0.024482
Outage_sec_perweek      0.004758
Bandwidth_GB_Year       0.000364
MonthlyCharge          -0.023967
Yearly_equip_failure   -0.024662
Name: 4, dtype: float64

Component 5 explains 5% of the data with these features contributing the most info:
  Contacts                0.998595
Yearly_equip_failure    0.012329
Outage_sec_perweek      0.000636
Bandwidth_GB_Year      -0.000836
MonthlyCharge          -0.003908
Tenure                 -0.006719
Age                    -0.050927
Name: 5, dtype: float64

Component 6 explains 1% of the data with these features contributing the most info:
  Tenure                  0.714384
MonthlyCharge           0.035687
Contacts                0.003477
Yearly_equip_failure    0.003464
Outage_sec_perweek     -0.000573
Age                    -0.016506
Bandwidth_GB_Year      -0.698630
Name: 6, dtype: float64
```

The principal components that explain 86% of the data are as follows:

1. Outage seconds per week
2. Monthly Charge
3. Yearly Equipment failure

In [73]:

E4. Describe how you identified the principal components of the data set.

Using Sklearn's PCA and a subset of the dataframe with continuous variables, I was able to use a cumaltive some bar chart and screeplot to visualize component variance explanation of the components. From there, I was able to use a function to read out the explanation of variance that a component has to the entire dataset as well as the features in each component in how they contribute to variance in the data.

5. Describe how the organization can benefit from the results of the PCA

PCA can help identify the numeric variables that account the most for variance in the data. The primary benefit is reducing dimensions of large datasets without losing too much information from a column. Reducing dimensions allows for the dataset to be less computationally expensive while still being accurate/

Panopto

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8083c54f-1d21-412b-a48b-aeca002ff6bb

References

IBM. (2021, August 17). Imputing or Filling Missing Values. Retrieved from IBM SPSS Documentation: ibm.com/docs/en/spss-modeler/SaaS?topic=values-imputing-filling-missing

Kumar, A. (2021, October 3). Python – Replace Missing Values with Mean, Median & Mode. Retrieved from Vitaflux: https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/

Yıldırım, S. (2020, March 2). A Practical Guide for Data Analysis with Pandas. Retrieved from Toward Data Science: https://towardsdatascience.com/a-practical-guide-for-data-analysis-with-pandas-e24e467195a9

In [78]:
```
pip install nbconvert --upgrade
```
```
Requirement already satisfied: nbconvert in c:\users\smcgb\anaconda3\lib\site-packages (6.4.4)
Collecting nbconvert
  Using cached nbconvert-6.5.0-py3-none-any.whl (561 kB)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (0.4)
Requirement already satisfied: beautifulsoup4 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (4.11.1)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (2.0.1)
Requirement already satisfied: nbformat>=5.1 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (5.3.0)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (4.9.2)
Collecting tinycss2
  Using cached tinycss2-1.1.1-py3-none-any.whl (21 kB)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: defusedxml in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: jupyterlab-pygments in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (0.1.2)
Requirement already satisfied: traitlets>=5.0 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (5.1.1)
Requirement already satisfied: bleach in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (4.1.0)
Requirement already satisfied: packaging in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (21.3)
Collecting jinja2>=3.0
  Using cached Jinja2-3.1.2-py3-none-any.whl (133 kB)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (1.5.0)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (0.5.13)
Requirement already satisfied: pygments>=2.4.1 in c:\users\smcgb\anaconda3\lib\site-packages (from nbconvert) (2.11.2)
Requirement already satisfied: pywin32>=1.0 in c:\users\smcgb\anaconda3\lib\site-packages (from jupyter-core>=4.7->nbconvert) (302)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\smcgb\anaconda3\lib\site-packages (from nbclient>=0.5.0->nbconvert) (6.1.12)
Requirement already satisfied: nest-asyncio in c:\users\smcgb\anaconda3\lib\site-packages (from nbclient>=0.5.0->nbconvert) (1.5.5)
Requirement already satisfied: tornado>=4.1 in c:\users\smcgb\anaconda3\lib\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (6.1)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\smcgb\anaconda3\lib\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: pyzmq>=13 in c:\users\smcgb\anaconda3\lib\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (22.3.0)
Requirement already satisfied: fastjsonschema in c:\users\smcgb\anaconda3\lib\site-packages (from nbformat>=5.1->nbconvert) (2.15.1)
Requirement already satisfied: jsonschema>=2.6 in c:\users\smcgb\anaconda3\lib\site-packages (from nbformat>=5.1->nbconvert) (4.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\users\smcgb\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\smcgb\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (21.4.0)
Requirement already satisfied: six>=1.5 in c:\users\smcgb\anaconda3\lib\site-packages (from python-dateutil>=2.1->jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (1.16.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\smcgb\anaconda3\lib\site-packages (from beautifulsoup4->nbconvert) (2.3.1)
Requirement already satisfied: webencodings in c:\users\smcgb\anaconda3\lib\site-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\smcgb\anaconda3\lib\site-packages (from packaging->nbconvert) (3.0.4)
Installing collected packages: tinycss2, jinja2, nbconvert
  Attempting uninstall: jinja2
    Found existing installation: Jinja2 2.11.3
    Uninstalling Jinja2-2.11.3:
      Successfully uninstalled Jinja2-2.11.3
  Attempting uninstall: nbconvert
    Found existing installation: nbconvert 6.4.4
    Uninstalling nbconvert-6.4.4:
      Successfully uninstalled nbconvert-6.4.4
Successfully installed jinja2-3.1.2 nbconvert-6.5.0 tinycss2-1.1.1
Note: you may need to restart the kernel to use updated packages.
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
spyder 5.1.5 requires pyqt5<5.13, which is not installed.
spyder 5.1.5 requires pyqtwebengine<5.13, which is not installed.
anaconda-project 0.10.2 requires ruamel-yaml, which is not installed.
jupyter-server 1.13.5 requires pywinpty<2; os_name == "nt", but you have pywinpty 2.0.2 which is incompatible.
```
In [76]:
```
!jupyter-nbconvert --to pdfviahtml WGUd206.ipynb
```
```
[NbConvertApp] Converting notebook WGUd206.ipynb to pdfviahtml
[NbConvertApp] ERROR | Notebook JSON is invalid: data must be valid exactly by one of oneOf definition

Failed validating <unset> in notebook:

On instance:
<unset>
[NbConvertApp] Writing 1436274 bytes to WGUd206.pdf
```