**D206 Task 1: Data Cleaning**

**Joseph Duszynski**

**ID #011237236**

**A. Research question**

Analysis of this data can be guided by determining if customer churn, data usage, and monthly payments can be predicted by the variables present in this dataset. The organization would benefit from knowing which customers are at risk of terminating services, what factors are correlated to higher customer payments, and the prediction of future customer data usage habits.

**B. Required variables**

1. Unnamed: 0. Numeric (int64). Sequential identifying column (identical to CaseOrder). Example: ‘15’.
2. CaseOrder. Numeric (int64). Sequential identifying column to preserve row ordering of original data file. Example: ‘232’.
3. Customer\_id. Categorical (object/string). Unique identifying string for each customer. Example: ‘V847470’.
4. Interaction. Categorical (object/string). Unique IDs pertaining to company interactions with customers. Example: ‘5c3548de-4744-4167-9412-3f018262bab2’.
5. City. Categorical (object/string). The city the customer resides in. Example: ‘Detroit’.
6. State. Categorical (object/string). The postal code of the state the customer lives in. Example: ‘TX’.
7. County. Categorical (object/string). State county of customer’s residence. Example: ‘Monroe’.
8. Zip. Numeric (int64). Customer’s zip code. Example: ‘50562’.
9. Lat. Numeric (float64). Customer’s latitude. Example: ‘35.43313’.
10. Lng. Numeric (float64). Customer’s longitude. Example: ‘35.43313’.
11. Population. Numeric (int64). Population within a one mile radius of the customer’s location. Example: ‘17351’.
12. Area. Categorical (object/string). Type of area customer lives in (urban, rural, or suburban). Example: ‘Urban’.
13. Timezone. Categorical (object/string). Customer’s time zone. Example: ‘America/New\_York’.
14. Job. Categorical (object/string). Customer’s self-reported job. Example: 'General practice doctor'.
15. Children. Numeric (float64). Number of children in the customer’s home. Example: '2.0'.
16. Age. Numeric (float64). Customer’s age. Example: '56.0'.
17. Education. Categorical (object/string). Customer’s highest earned degree or school diploma. Example: 'Regular High School Diploma'.
18. Employment. Categorical (object/string). Customer’s employment status. Example: 'Full Time'.
19. Income. Numeric (float64). Customer’s income per year. Example: ‘115114.57’.
20. Marital. Categorical (object/string). Customer’s marital status. Example: 'Widowed'.
21. Gender. Categorical (object/string). Customer’s gender. Example: 'Male'.
22. Churn. Categorical (object/string). Whether or not the customer terminated services with the company in the past month. Example: 'Yes'.
23. Outage\_sec\_perweek. Numeric (float64). The average outage time near the customer’s residence, measured in seconds per week. Example: '13.66394795'.
24. Email. Numeric (int64). Number of emails sent to the customer by the company over the past year. Example: '10'.
25. Contacts. Numeric (int64). The number of occasions the customer has contacted technical support. Example: '3'.
26. Yearly\_equip\_failure. Numeric (int64). The number of times a customer’s equipment had to be reset or replaced within the past year. Example: '0'.
27. Techie. Categorical (object/string). Customer’s self-reported assessment of their technical ability. Example: 'Yes'.
28. Contract. Categorical (object/string). Customer’s contract term (frequency). Example: 'Month-to-month'.
29. Port\_modem. Categorical (object/string). Whether or not the customer possesses a portable modem. Example: 'Yes'.
30. Tablet. Categorical (object/string). Whether or not the customer has a tablet. Example: 'No'.
31. InternetService. Categorical (object/string). The type of internet service provider the customer uses (‘None’ if they don’t have one). Example: 'DSL'.
32. Phone. Categorical (object/string). Whether or not the customer utilizes a phone line. Example: 'Yes'.
33. Multiple. Categorical (object/string). Whether or not the customer has multiple lines. Example: 'Yes'.
34. OnlineSecurity. Categorical (object/string). Whether or not the customer uses an online security add-on. Example: 'Yes'.
35. OnlineBackup. Categorical (object/string). Asks if the customer utilizes an online backup add-on. Example: 'Yes'.
36. DeviceProtection. Categorical (object/string). Asks if the customer utilizes a device protection add-on. Example: 'Yes'.
37. TechSupport. Categorical (object/string). Asks if the customer utilizes a technical support add-on. Example: 'Yes'.
38. StreamingTV. Categorical (object/string). Asks if the customer’s TV has streaming capabilities. Example: 'Yes'.
39. StreamingMovies. Categorical (object/string). Asks if the customer can stream movies. Example: 'Yes'.
40. PaperlessBilling. Categorical (object/string). Whether or not the customer uses paperless billing. Example: 'Yes'.
41. PaymentMethod. Categorical (object/string). The type of payment method used by the customer. Example: 'Credit Card (automatic)'.
42. Tenure. Numeric (float64). How long (in months) the customer has been with the cable provider. Example: '6.649321'.
43. MonthlyCharge. Numeric (float64). The average monthly amount billed to the customer. Example: '129.5085'.
44. Bandwidth\_GB\_Year. Numeric (float64). The gigabytes of data used by the customer per year. Example: '5899.518'.
45. item1. Numeric (int64). Survey question 1 of 8 (“Timely response”) on a scale of 1 (most important) to 8 (least important). Example: '5'.
46. item2. Numeric (int64). Survey question 2 of 8 (“Timely fixes”) on a scale of 1 (most important) to 8 (least important). Example: '3'.
47. item3. Numeric (int64). Survey question 3 of 8 (“Timely replacements”) on a scale of 1 (most important) to 8 (least important). Example: '3'.
48. item4. Numeric (int64). Survey question 4 of 8 (“Reliability”) on a scale of 1 (most important) to 8 (least important). Example: '2'.
49. item5. Numeric (int64). Survey question 5 of 8 (“Options”) on a scale of 1 (most important) to 8 (least important). Example: '4'.
50. item6. Numeric (int64). Survey question 6 of 8 (“Respectful response”) on a scale of 1 (most important) to 8 (least important). Example: '6'.
51. item7. Numeric (int64). Survey question 7 of 8 (“Courteous exchange”) on a scale of 1 (most important) to 8 (least important). Example: '3'.
52. item8. Numeric (int64). Survey question 8 of 8 (“Evidence of active listening”) on a scale of 1 (most important) to 8 (least important). Example: '2'.

**C. 1. Plan to assess quality of data**

The objectives for assessing data quality consist of detecting duplicates, outliers, null values, and improper or impossible values violating constraints within a column (such as a negative number of emails).

First, get an overview of the data through commands like .info() and .describe(), as well as initial detection of nulls and duplicates with df.isna().sum() > 0 and df.duplicated().sum(), respectively. Using the method display\_value\_counts, each column’s number of null values, number of unique values, and its .value\_counts(dropna=False) output provide initial insight into which columns contain null values, extreme outliers, or other anomalies that might violate constraints (such as a column expecting positive integers having a minimum less than 0). It’s also important to investigate redundant columns such as ‘Unnamed: 0’ and ‘CaseOrder’. They appear identical but this can be verified through (df['Unnamed: 0'] == df['CaseOrder']).sum().

Searching for outliers using the interquartile range method is done with outlier\_search, which outputs the IQR bounds and any offending rows outside of those thresholds.

Any constraint violations observed from the output of the above methods can prompt individual investigation on those rows and columns. Specifically for this dataset, counting and returning rows where the zip code isn’t five digits as well as those where the average weekly outage is negative.

Producing histograms of various columns, histograms of their z-scores, and a series of histograms (null\_histogram\_gen) on a given column when rows containing nulls in other columns are removed or excluded (e.g. histogram of ‘Phone’ when rows with null ‘Income’ are removed compared to a histogram of ‘Phone’ when only rows with null ‘Income’ are included) can identify possible relationships and patterns in the null values.

For a more thorough investigation of duplicates, I used the function n\_column\_duplicate\_tester. It searches for duplicates across a subset of columns that must have n values in common between two or more rows to be flagged as a potential duplicate. There are additional optional parameters that can require one or more columns to always be included in the searches (they’re added on top of the n columns shared in common) and a setting that bounds how many of the shared columns can have null values. Rows detected as possible duplicates based on mostly or only null values may or may not be duplicates, but shared non-null values would distinguish or unite them. A set of duplicate rows to match on such as city, job, age, income, tenure, and others, with n = 3, n = 4, and n = 5, was used to locate and inspect any near duplicates that wouldn’t be caught with a .duplicated(subset=columns, keep=False) approach.

Validator methods such as cross\_validator and agg\_validator were written to scan for anomalies. In particular, agg\_validator was used to detect unusually high aggregate standard deviations of latitude and longitude when the data was grouped by state and county (a unique identifier), expecting a relatively limited set of valid latitude and longitude values unless the county was unusually large.

MinHashing was used to search for near-duplicate (redundant) values in categorical columns with a large number of unique values that would be too cumbersome to manually look over. Specifically, this applied to the ‘Job’ column in this data set, as the only other categorical columns with a multitude of unique values were geographical. Initial MinHashing found several instances of duplicated job titles such as ‘Accountant, chartered’ and ‘Chartered accountant’. This prompted three different string transforms that created three additional variant job columns (‘Job\_filt’, ‘Job\_filt\_inv’, and ‘Job\_hierarch’ – see section D.5. for additional details).

Several methods were used to look for correlations in null values. The function null\_inspector provided summary statistics on the columns containing nulls across a chosen set of (numerical) comparison columns. Using chosen thresholds in percent differences for the mean and standard deviation (difference taken between mean and standard deviation when the null column’s nulls were excluded vs when its non-nulls were excluded), it compiled a list of candidate correlations between one of the columns containing null values and one of the columns among the set of comparison columns.

Following the numerical inspection, missingno dendrograms, heatmaps, and a missingness matrix were used to search for possible patterns, as well as the seaborn pairplot. In addition, scatterplots where the nulls were filled in with dummy values and isolated on the margins were utilized to look for any aberrations.

A number of approaches were adopted to scan for constraint violations and potential subtle errors that wouldn’t have been recognized by the previous work. The high frequency of population values of 0, especially in urban areas where other customers living nearby had population values in the hundreds or thousands, required investigation by isolating the dataframe to a low threshold value (20), producing histograms, and creating an aggregate dataframe on city and state, to be discussed in greater detail in section D.1. The distribution of ‘Bandwidth\_GB\_Year’ for customers having no internet, customers where their annual billing statements exceed 50 percent of their income, and strange distribution of incomes seemingly independent of employment status and career (e.g. a full time psychologist making less than 10000 per year) also required manual inspection of subsets of dataframes and histograms. Sub-dataframes on the basis of ‘Area’ category were created to search for relationships and anomalies, such as any impact it has on the average weekly outage vs average annual bandwidth scatterplot and frequency of low populations (which are expected to be concentrated in rural areas). This is discussed in greater detail in comments around line 734 in “data\_assessment.py”.

Finally, the previously identified linear relationship between ‘Tenure’ and ‘Bandwidth\_GB\_Year’ permitted a linear regression model with a very high r2 value. The model’s parameters can be used later for estimating null values.

**C. 2. Justification of approach**

As discussed in the previous section, the objective is to identify duplicates, outliers, null values, and improper values violating constraints imposed by the column or reality (i.e. entered in error).

For duplicates, initial usage of .duplicated() didn’t find anything obvious, but a more complete verification would search for matches on at least a few different columns, likely sharing a value such as ‘City’ (and state) at a minimum. Automating this required the function n\_column\_duplicate\_tester to scan across all (len(candidate\_columns) choose n) combinations. Near duplicate strings in a column’s unique values (limited to the ‘Jobs’ column here) necessitated using a string comparison method, which was done most efficiently with a fuzzy hashing approach.

Histograms, interquartile ranges, and summary statistics were best suited to identify outliers. outlier\_search, zscore\_hists, and plots\_by\_column were written to provide all of that information while reviewing the overall distribution to gauge if the outliers were entered in error or simply a result of the variable’s natural distribution.

Specifying null values and their locations is quickly done with .isnull() manipulations, but knowing after detection there will be imputation (or some form of modification), it was important to try various methods to search for patterns and correlations in the missing values (i.e. determine if a column’s null values are MCAR, MAR, or MNAR). Histograms created through null\_histogram\_gen and other functions allowed for a visual inspection of significant changes in proportions (on categorical variables) when rows with null values in a specific column were excluded vs excluding non-null values. Numerical comparisons were done with null\_inspector, which offered greater detail and output. Additionally, missingno plots, the seaborn pairplot, and scatterplots with dummy values (adjusted\_null\_scatter) provided insights into the structure of the data and its missing values.

Lastly, values violating constraints or otherwise entered in error required improvisations directed by the investigations into correlations, nulls, and outliers. Methods like .value\_counts() can quickly confirm everything is sensible in categorical columns with a small set of unique values (e.g. only ‘Yes’ or ‘No’ in a binary column, no typos), but are of little help if a systematic error or lack of updating leaves unusual discrepancies that aren’t large enough to be detected with histograms and outlier scanning nor trivial enough to write off as noise. display\_value\_counts assisted in discovering negative outage values and improperly formatted zip codes. cross\_validator and agg\_validator could be applied generally across many subsets of columns (beyond cross-validating geographical data), although determining threshold values can be tricky. The seaborn pairplot was quite helpful in identifying clustering patterns and observing the unusual lack of points around the means of ‘Tenure’ and ‘Bandwidth\_GB\_Year’, indicating many of the null values in those columns might correspond to those ranges (given their strong linear relationship). Histograms and reviewing sections of dataframes from everything mentioned in these past two sections found unexpected patterns that are difficult to explain as natural but not strong enough to conclude the data was entered incorrectly. This again pertains to the low values for income and population that don’t create an unnatural distribution but are hard to believe as accurate. This will be discussed in greater detail in sections below this one. Ultimately, discovering subtle sources of error can be rather difficult and you’re never guaranteed to be absolutely sure you’ve found all inaccurate entries. Idiosyncratic inspections and discoveries, potentially occurring on accident, require a thorough search for correlations and use of external knowledge (“common sense”).

**C. 3. Justification of tools**

I chose Python 3.9 for its ease of use, widespread adoption, personal familiarity, speed, multitude of mathematical and statistical packages, and extensive documentation and examples online for debugging and understanding. I didn’t need intricate or beautiful images and prefer the consistent syntax of Python and its packages compared to libraries in R. As an added benefit, I can utilize the functions written in this project for future work.

Numpy and pandas were used for handling numerical computations on arrays and dataframes, respectively. They’re effectively mandatory for any of the project to work. I needed matplotlib and seaborn to produce histograms, scatterplots, and pairplots. Scipy.stats was required for calculating z-scores to be used in creating z-score histograms. Missingno was used for its missingness matrix, dendrogram, and heatmap to review possible relationships between null values. pdb was used to create natural breakpoints as the code runs while copy was used to create deep copies of arrays. From scikit-learn, linear\_model was used for a linear fit on ‘Tenure’ and ‘Bandwidth\_GB\_Year’ to assist in imputation. Later, I used IterativeImputer for a MICE imputation model, KNNImputer to compare it with a KNN model, StandardScaler and train\_test\_split to normalize and prepare the data before using the modeling functions, and model metrics of r2\_score, accuracy\_score, and mean\_squared\_error to evaluate goodness of fit. PCA from sklearn.decomposition was used for the required PCA. Imputing values in the null categorical columns of ‘Phone’, ‘Techie’, and ‘TechSupport’ was very unreliable with most models because of the imbalances present, so BalancedRandomForestClassifier from imblearn.ensemble helped produce decent enough models for imputations on those columns. To generate all possible (t choose n) combinations in n\_column\_duplicate\_tester, combinations from itertools was used. MinHash and MinHashLSH from datasketch were needed to perform an efficient comparison of near-duplicate strings when filtering the ‘Job’ column.

**C. 4. Data assessment code**

See attached file “data\_assessment.py” and text file version “data\_assessment.txt”.

**D. 1. Cleaning findings**

1. Duplicates:
   1. By row, no duplicate entries were found with .duplicated(). Using n\_column\_duplicate\_tester, four candidates (two pairs of two) were found sharing the same zip code, city, and job. They differed on gender, marital status, age, and other variables, so were false positives. For city, zip code, and age, there were 32 matches (sharing non-null values on all three columns). Individual inspection of each of them found sufficient differences in other columns to verify they weren’t duplicated. Remaining combinations contained null values and were discarded as false positives (e.g. sharing a job and age but living in a different city). n = 4 and n = 5 produced no matches at all. With high confidence, there are no duplicated rows in this dataset.
   2. Within the ‘Job’ column, consolidation performed by job\_inspect reduced an initial set of 639 unique ‘Job’ strings to 517 in ‘Job\_filt’ and ‘Job\_filt\_inv’ (519 in ‘Job\_hierarch’). There is some grey area in distinguishing between ‘Accountant, chartered’, ‘Accountant, chartered certified’, and ‘Accountant, chartered management’. In the interest of data integrity, those were left alone for the purposes of this project. The filtering reduction ultimately identified 122 duplicate values for ‘Job’ and unified them under one of the formats described in section D.5.
2. Outliers using IQR method:
   1. For columns 'item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8':
      1. ‘item1’ had an IQR lower bound of 1.5 and an upper bound of 5.5, finding 442 outliers. The minimum was 1.0 and the maximum was 7.0 with the 75th percentile being 4.0.
      2. ‘item2’ had an IQR lower bound of 1.5 and an upper bound of 5.5, finding 445 outliers. The minimum was 1.0 and the maximum was 7.0 with the 75th percentile being 4.0.
      3. ‘item3’ had an IQR lower bound of 1.5 and an upper bound of 5.5, finding 418 outliers. The minimum was 1.0 and the maximum was 8.0 with the 75th percentile being 4.0.
      4. ‘item4’ had an IQR lower bound of 1.5 and an upper bound of 5.5, finding 433 outliers. The minimum was 1.0 and the maximum was 7.0 with the 75th percentile being 4.0.
      5. ‘item5’ had an IQR lower bound of 1.5 and an upper bound of 5.5, finding 422 outliers. The minimum was 1.0 and the maximum was 7.0 with the 75th percentile being 4.0.
      6. ‘item6’ had an IQR lower bound of 1.5 and an upper bound of 5.5, finding 413 outliers. The minimum was 1.0 and the maximum was 8.0 with the 75th percentile being 4.0.
      7. ‘item7’ had an IQR lower bound of 1.5 and an upper bound of 5.5, finding 454 outliers. The minimum was 1.0 and the maximum was 7.0 with the 75th percentile being 4.0.
      8. ‘item8’ had an IQR lower bound of 1.5 and an upper bound of 5.5, finding 426 outliers. The minimum was 1.0 and the maximum was 8.0 with the 75th percentile being 4.0.
   2. For columns 'Children', 'Age', 'Email', 'Contacts', 'Yearly\_equip\_failure':
      1. ‘Children’ had an IQR lower bound of -4.5 (bounded to 0 in practice) and an upper bound of 7.5, finding 302 outliers. The minimum was 0.0 and the maximum was 10.0 with the 75th percentile being 3.0.
      2. ‘Age’ had an IQR lower bound of -19.0 (bounded to 0 in practice) and an upper bound of 125, finding 0 outliers. The minimum was 18.0 and the maximum was 89.0 with the 75th percentile being 71.0.
      3. ‘Email’ had an IQR lower bound of 4.0 and an upper bound of 20.0, finding 38 outliers. The minimum was 1.0 and the maximum was 23.0 with the 75th percentile being 14.0.
      4. ‘Contacts’ had an IQR lower bound of -3.0 (bounded to 0 in practice) and an upper bound of 5.0, finding 8 outliers. The minimum was 0.0 and the maximum was 7.0 with the 75th percentile being 2.0.
      5. 'Yearly\_equip\_failure' had an IQR lower bound of -1.5 (bounded to 0 in practice) and an upper bound of 2.5, finding 94 outliers. The minimum was 0.0 and the maximum was 6.0 with the 75th percentile being 1.0.
   3. For columns 'Lat', 'Lng', 'Population', 'Income', 'Outage\_sec\_perweek', 'Tenure', 'MonthlyCharge', 'Bandwidth\_GB\_Year':
      1. ‘Lat’ had an IQR lower bound of 25.194 and an upper bound of 52.255, finding 158 outliers. The minimum was 17.966 and the maximum was 70.641 with the 75th percentile being 42.107.
      2. ‘Lng’ had an IQR lower bound of -122.574 and an upper bound of -54.598, finding 273 outliers. The minimum was -171.688 and the maximum was -65.668 with the 75th percentile being -80.089.
      3. ‘Population’ had an IQR lower bound of -17907.0 (bounded to 0 in practice) and an upper bound of 31813.0, finding 937 outliers. The minimum was 0.0 and the maximum was 111850 with the 75th percentile being 13168.0.
      4. ‘Income’ had an IQR lower bound of -31994.786 (bounded to 0 in practice) and an upper bound of 104752.704, finding 249 outliers. The minimum was 740.660 and the maximum was 258900.700 with the 75th percentile being 53472.395.
      5. ‘Outage\_sec\_perweek’ had an IQR lower bound of 1.404 and an upper bound of 19.138, finding 539 outliers. The minimum was -1.349 and the maximum was 47.049 with the 75th percentile being 12.488.
      6. ‘Tenure’ had an IQR lower bound of -72.414 (bounded to 0 in practice) and an upper bound of 141.731, finding 0 outliers. The minimum was 1.000 and the maximum was 71.999 with the 75th percentile being 61.427.
      7. ‘MonthlyCharge’ had an IQR lower bound of 47.012 and an upper bound of 297.837, finding 5 outliers. The minimum was 77.505 and the maximum was 315.879 with the 75th percentile being 203.777.
      8. ‘Bandwidth\_GB\_Year’ had an IQR lower bound of -5295.368 (bounded to 0 in practice) and an upper bound of 12116.575, finding 0 outliers. The minimum was 155.507 and the maximum was 7158.982 with the 75th percentile being 5587.097.
3. Null values:
   1. ‘Children’ had 2495 null values.
   2. ‘Age’ had 2475 null values.
   3. ‘Income’ had 2490 null values.
   4. ‘Techie’ had 2477 null values.
   5. ‘InternetService’ had 2129 null values upon initial detection. However, “None” is a valid entry, which is detected as a null value upon importing the data into Python. The null values in this column are appropriately filled with the string “None”.
   6. ‘Phone’ had 1026 null values.
   7. ‘TechSupport’ had 991 null values.
   8. ‘Tenure’ had 931 null values.
   9. ‘Bandwidth\_GB\_Year’ had 1021 null values.
4. Ignoring constraints and incorrect (non-null) entries:
   1. ‘Zip’ is stored as an integer so any entry with leading zeros becomes incorrectly formatted. 773 of them need to be adjusted.
   2. ‘Outage\_sec\_perweek’ has 11 negative values when its minimum should be 0.00.
   3. Many ‘Population’ values were 0, frequently residing in urban areas or close to other customers who had significantly higher values for ‘Population’. There are 150 customers with recorded values for ‘Population’ less than 20.
   4. Customers with ‘InternetService’ = “None” have nonzero bandwidth values that are in fact comparable to those with internet service.
   5. ‘Multiple’ seems like it should always be ‘No’ when ‘Phone’ = ‘No’, but seems to have little relationship to the value of ‘Phone’.
   6. The annual charge (12 \* ‘MonthlyCharge’) amount on several occasions is a significant share of the customer’s annual income if not exceeding it. There are 129 instances where the annual charge exceeds 50 percent of the customer’s income.
   7. There are surprisingly low and high incomes that appear to be independent of employment status or usual income range associated with the customer’s listed job title. Full time psychologists making less than 10000 per year for example.

**D. 2. Justification of mitigation methods**

1. Duplicates:
   1. As discussed in D.1.1.a., no duplicate entries by row were found, so nothing needed to be done.
   2. For redundant (duplicated) values in the ‘Job’ column, should there be some unknown nuance or reasoning behind the redundancy (perhaps it’s input by the customer and it would be preferable for them to access the same string they recall entering), the original ‘Job’ column was untouched for the time being. For organizational purposes, depending on what’s preferred, alternate columns (‘Job\_filt’, ‘Job\_filt\_inv’, and ‘Job\_hierarch’) were created. As described in section D.5., ‘Job\_filt’ consolidates jobs of the form ‘B, A’ into ‘A B’ entries (no more commas). ‘Job\_filt\_inv’ inverses this procedure and instead retains ‘B, A’ for any entry of the form ‘A B’ under the condition that ‘B, A’ occurred at least once in ‘Job’. ‘Job\_hierarch’ generalizes the categorization as much as possible, attempting to keep jobs under the same broad category (such as ‘Engineer’, ‘Researcher’, and so on) with the category first when alphabetically sorting (i.e. ‘category, subfield’ format whenever it’s identified).
2. Outliers:
   1. ‘item1’ through ‘item8’: For a survey of eight questions with no requirement that the integer scores of 1 through 8 only be used once, it’s to be expected that many customers will consider most if not all of the aspects within the questions to be of at least moderate importance. Every single question consistently had an IQR lower bound of 1.5, upper bound of 5.5, mean of approximately 3.5, minimum of 1.0, and maximum of either 7.0 or 8.0. While values of 6.0 or above are infrequent, looking at the histograms and z-scores, their rare proportions are expected and almost certainly genuine customer input. As all values are within the expected range [1.0, 8.0], this is a positively skewed distribution that needs no intervention or adjustments, as that would distort values that by all counts appear to be accurate.
   2. For columns 'Children', 'Age', 'Email', 'Contacts', 'Yearly\_equip\_failure':
      1. ‘Children’: Reviewing the histogram, the number of children declines as expected. The offending values aren’t significantly higher than the IQR upper bound nor are there that many of them. As they’re likely genuine and the difference is small, it’s best to keep them.
      2. ‘Age’: No outliers or negative values, so there’s nothing to be done.
      3. ‘Email’: The upper bound was 20 and the maximum was 23, with a small minority of outliers. Reviewing the histogram, all of this appears to be genuine, and the values are barely outside of the IQR threshold. It’s best that they are retained.
      4. ‘Contacts’: An upper bound of 5 with a maximum of 7 and only 8 outliers. These are likely genuine values. As they’re barely above the threshold and there are so few of them, their presence makes little difference. It’s best to keep them as they are.
      5. ‘Yearly\_equip\_failure’: Approximately 1 percent of customers seem to have extraordinarily bad luck. The upper IQR bound of 2.5 and reviewing the histogram demonstrate the vast majority of customers have little to no trouble with their equipment. An analyst might wish to study this subpopulation in greater detail, and since this looks like a genuine bimodal distribution, it would be detrimental to remove or adjust them. Customers facing significantly high failure rates could be in an unfortunate region or all share the same technician. Retaining these genuine values is better than discarding them or altering them.
   3. For columns 'Lat', 'Lng', 'Population', 'Income', 'Outage\_sec\_perweek', 'Tenure', 'MonthlyCharge', 'Bandwidth\_GB\_Year':
      1. ‘Lat’: A scatter plot with ‘Lng’ shows the geographic distribution of the continental U.S. vs Alaska, Hawaii, and Puerto Rico. Understandably, areas far from the mainland U.S. might be flagged as outliers. The IQR method is inadequate here and its declared outliers should remain as they are.
      2. ‘Lng’: Similar to the above, residents in Alaska and Hawaii are genuine, not outliers or errors.
      3. ‘Population’: The outliers detected with IQR reflect the wide variance in secluded rural areas compared to large, densely packed cities. The histogram shows a positively skewed distribution retaining its shape at different scales (i.e. isolating to populations above 20000 shows the same overall shape) that rapidly extinguishes at the outlier populations. Reviewing those rows within the dataframe, they do correspond to populated areas and appear to be genuine. Another aspect to keep in mind is the inaccurately low values of population in many entries. The 0 values in urban areas are most obvious, but as discussed elsewhere the unusual inconsistencies in population, along with few points of comparison (the density of (city, state) combinations is rather sparse), require customer follow-up and geographical cross-validation that’s outside of the scope of this project. For the time being, these high population values appear to be accurate and will be left untouched.
      4. ‘Income’: The histogram of this data reflects modern income distributions. The larger values are almost certainly accurate entries in the system. However, there is a broader problem (see section D.2.4.g.) with discrepancies in ‘Income’, ‘Employment’, and ‘Job’. This will invariably require follow-up with customers. For the time being, the high values outside of the IQR will be retained as they are believed to be genuine nor do they distort the analysis.
      5. ‘Outage\_sec\_perweek’: The clustered nature of outages between storms, seasonal impacts, and areas that have poor infrastructure and face more frequent outages than other regions, all combine to make this an inadequate metric. Outliers would be more meaningful if it were recorded over a much shorter period of time, perhaps a day or two. When outages can last for hours or days, a maximum of 47.05 implies this value is averaged over a very long period of time. The histogram shows a bimodal distribution where the majority of the population is clustered under ~10. The distribution of the second mode indicates it’s genuine and not the result of incorrect input. While it might be useful to study the different clusters, the ~5 percent of customers in the second cluster are an important component to the analysis that shouldn’t be excluded. As the values appear to be accurate, they won’t be modified.
      6. ‘Tenure’: Zero outliers detected and the only point of note on its histogram is the bimodal distribution that oddly has very few points around the mean. As discussed previously, many of the null values could belong to this area. Regardless, without any outliers, nothing needs to be altered under ‘Tenure’ in this respect at least.
      7. ‘MonthlyCharge’: As there were only five outliers that barely exceeded the IQR bound of 297, they’re likely accurate nor do they distort any analysis. It’s best to leave them alone.
      8. ‘Bandwidth\_GB\_Year’: No outliers found with IQR and the histogram shows a bimodal distribution. Nothing to be altered here.
3. Null values:
   1. ‘InternetService’: As mentioned in D.1.3.e., the 2129 null values in ‘InternetService’ resulted from Python regarding valid “None” strings as nulls. Counting “None” values using a spreadsheet program verified there were no alternative null forms in this column (such as blank cells or “NA”). This was corrected by using df\_churn['InternetService'] = df\_churn['InternetService'].fillna("None").
   2. ‘Tenure’ and ‘Bandwidth\_GB\_Year’: Inspecting the seaborn pairplot showed a strong linear relationship between ‘Tenure’ and ‘Bandwidth\_GB\_Year’. A linear regression model, tenure\_band\_lin\_fit(), was used above a threshold value of 1000 for ‘Bandwidth\_GB\_Year’ as the data became sparse and noisy below this value. The parameters of the resulting fit were used to calculate the associated ‘Tenure’ or ‘Bandwidth\_GB\_Year’ when at least one of them was non-null and the ‘Bandwidth\_GB\_Year’ value (observed) was above the cutoff of 1000. If the fit had been done across all bandwidth values, it would calculate negative values of ‘Tenure’. Between rows that had null entries for both tenure and bandwidth or had null tenure with a bandwidth below the cutoff, the nearly 2000 affected rows had been reduced to 237 points which would require a different imputation approach.
   3. ‘Children’, ‘Age’, ‘Income’: For the null values in ‘Children’, ‘Age’, and ‘Income’, no human-discernible relationships were found through the investigations mentioned so far. Using MICE and KNN models and comparing the density plots, it was determined the KNN model most closely resembled the existing density plots (MICE strongly favored the mean). KNN imputations overwrote the null values in ‘Children’, ‘Age’, ‘Income’, as well as the remaining nulls in ‘Tenure’ and ‘Bandwidth\_GB\_Year’, ensuring ‘Children’ and ‘Age’ remained as integers and the maxima and minima in the affected columns were unaltered.
   4. ‘Phone’, ‘Techie’, ‘TechSupport’: These categorical columns that had binary yes/no responses did not appear to have any obvious relationship to other variables. Given the imbalanced proportions (e.g. ‘Phone’ has nearly 90 percent ‘Yes’), many classifier approaches were inadequate, but a balanced random forest with parameter tuning produced models that were as reasonably accurate as could be hoped for and produced a distribution similar to the original. Upon reaching accuracy scores as high as they could go without overfitting, rows with nulls in these columns received predictions from their respective models that would then replace the null values.
4. Ignoring constraints and incorrect (non-null) entries:
   1. ‘Zip’ was backed up to a new column ‘Zip\_int64’ while the existing column was cast as a string. Existing entries with a number of digits less than five were adjusted to have the appropriate leading 0s.
   2. As there were only 11 negative values in ‘Outage\_sec\_perweek’, with a minimum of -1.349, these were likely accidental inputs with a hyphen or some other means of adding a negative sign. Given the nontrivial frequency of low values around +1.0, the most appropriate action was to replace them with their absolute value.
   3. The function population\_adjuster created an aggregate dataframe of cities where the minimum ‘Population’ entry was less than 10 percent of the maximum. When a customer’s population was below a threshold of 20 and they had neighbors in the same city with significantly higher populations (specifically their population was less than 15 percent of the maximum), it was almost certainly an incorrect input that was then adjusted to the mean for that (city, state) combination. Of the 150 customers with populations under 20 listed, 43 of them were adjusted to their local means, while the other 107 had insufficient data to establish their accurate local populations. At the same time, customers in isolated areas could genuinely have a population of 0 within a one mile radius, so it was best to leave those values alone without additional information. For safety, the original values were backed up to a new column ‘Population\_raw’.
   4. The general lack of any relationship between ‘InternetService’ = “None” and average bandwidth usage indicates there’s a misunderstanding and that nothing is in error. For instance, they might have internet service through another provider but still report their used bandwidth. Nothing needed to be changed here.
   5. Similar to the above, ‘Phone’ = ‘No’ received about a 50 percent split of ‘Yes’ and ‘No’ for ‘Multiple’, so it was considered an accurate input and simple misunderstanding requiring no adjustments or interventions.
   6. As discussed around line 734 of “data\_assessment.py”, this would require further evaluation outside the scope of this project. Customers may be reporting their own income while receiving assistance from a household member in paying bills, whereas other customers might report household income. The incomes were generally rather low across the entire customer base despite the distribution displaying the expected skew pattern. At this stage, it would be imprudent to adjust incomes on the basis of the customer’s monthly billing statement being too high for them to afford it.
   7. The very low income values and seemingly inconsistent relationships to employment status and job title requires further follow-up with customers. Presumably those listed as 'Retired' retain their previous career under 'Job'. Widely varying and unexpected incomes, such as full time doctors and engineers with incomes under 10000, and retired customers having high incomes, could potentially be explained by a customer reporting individual vs household income, failing to update, under- or over-reporting, error in data entry, geographic location, pensions, and other reasons. The overall distribution of incomes skews low, but doesn't show signs of widespread systematic error. Investigating its accuracy would require external datasets or additional contact with customers that currently isn't possible within this project. For these reasons, 'Income', 'Job', and 'Employment' will be left alone (with the exception of KNN imputations for null values in 'Income') at the present.

**D. 3. Summary of the outcomes**

As the program runs, the first modification occurs with job\_inspect() and show\_frequent\_career\_cats(frequent\_job\_fields\_0, job\_list\_0). This creates three alternate columns ‘Job\_filt’, ‘Job\_filt\_inv’, and ‘Job\_hierarch’ that reduce the 639 unique values in ‘Job’ to 517 (519 in the case of ‘Job\_hierarch’). See section D.2.1.b. or D.2. for additional details.

Next, the misinterpreted ‘None’ values in ‘InternetService’ are filled with string values of “None”. Afterward, ‘Zip’ is backed up to ‘Zip\_int64’ (retaining its numeric datatype) and adjusted to a string format with leading 0s restored. Then the negative ‘Outage\_sec\_perweek’ values are replaced with their absolute values.

Using the parameters from the linear fit between ‘Tenure’ and ‘Bandwidth\_GB\_Year’, band\_tenure\_linear\_impute(band\_cut=cutoff\_bandwidth) fills in null values when the bandwidth is above the cutoff of 1000 and there’s at least one non-null value between them.

MICE and KNN imputation models are created and compared, determining KNN more accurately resembles the original distributions of the relevant variables. The null values in the numerical columns ‘Age’, ‘Children’, ‘Income’, as well as the remaining null values in ‘Tenure’ and ‘Bandwidth\_GB\_Year’, are replaced with the model’s predictions. The linear fit between ‘Tenure’ and ‘Bandwidth\_GB\_Year’ is run again (tenure\_band\_lin\_fit) to validate r2 remains high.

To correct anomalously low population values where possible, population\_adjuster() uses the aggregate dataframe filt\_agg\_df to replace population values below the threshold (and meeting the other conditions discussed in section D.2.4.c.) with the mean values of nearby customers. The original values were backed up to ‘Population\_raw’.

A balanced random forest classifier was used to make predictions for ‘Phone’, ‘Techie’, and ‘TechSupport’. The null values in those columns were replaced with their respective model’s predictions.

Finally, the cleaned data was reviewed for any remaining nulls, outliers, anomalies, or column constraint issues. After confirming it satisfied the project’s goals, it was saved to a new file. The data now has no null values, no duplicates, and all data points flagged as outliers were determined to accurate, genuine, and consistent. See the attached file “cleaned\_churn\_raw\_data.csv” discussed in section D.5.

**D. 4. Data cleaning (mitigation) code**

See attached file “data\_cleaning.py” and text file version “data\_cleaning.txt”. As noted in the beginning of the file, the data cleaning and modifications begin at line 825 (variant ‘Job’ columns is from line 410 to 558). The methods and variables used in part C. to detect issues with the data were also used in its cleaning, so they’re left untouched (above line 825).

**D. 5. Cleaned data**

See attached file “cleaned\_churn\_raw\_data.csv”. It retains the same columns and datatypes as before with the following exceptions:

1. ‘Zip’ was changed from numeric (int64) to categorical (string/object). The original values were backed up to a new column ‘Zip\_int64’ of type ‘int64’. This was done to ensure they followed the five digit format expected and more appropriately categorized as a categorical datatype.
2. Consolidating similar strings in the ‘Job’ column led to the creation of three alternate columns: ‘Job\_filt’, ‘Job\_filt\_inv’, and ‘Job\_hierarch’, all of which retain the categorical (string/object) datatype of their parent column ‘Job’. ‘Job\_filt’ replaces any job of the form ‘job\_field, sub\_field’ with ‘sub\_field job\_field’ (and appropriate capitalization), such as ‘Engineer, mechanical’ to ‘Mechanical engineer’. ‘Job\_filt\_inv’ inverses this process to remove duplicate instances of ‘B, A’ and ‘A B’ in the original ‘Job’ column (choosing to keep the ‘B, A’ version). For ‘Job\_hierarch’, a list of frequently occurring substring career fields (e.g. ‘Engineer’, ‘Therapist’) was formed to create a more logical and organized selection of jobs. Frequently occurring career fields that might generally occur at the end of the string were moved to the front and given a comma to produce as many instances of ‘job\_field, sub\_field’ as possible. This maintains a better organization and alphabetical ordering.
3. Low values of ‘Population’ below a threshold of 20 that were also less than 15 percent of the maximum recorded for other entries in the same city (where applicable) were replaced with the mean for entries in that (‘City’, ‘State’). The old values were backed up to ‘Population\_raw’. Both columns retain the original numeric (int64) datatype.

**D. 6. Limitations**

This data cleaning process is inherently imperfect and limited. Null values can easily be identified, but imputation procedures are ultimately estimates. For instance, tenure and bandwidth have a very strong linear correlation, a similar number of null values, and a conspicuously absent cluster of data around their means. Ostensibly the null values might fill in this cluster, but using the parameters from a linear fit to predict the corresponding tenure or bandwidth value only partially fills in the vacant regions. With the data presented, we can only speculate why customers either drop off at that point or the company had problems getting new customers some months back. The balanced random forest classifier was an improvement over mode imputation, but in the case of ‘TechnicalSupport’ it had model accuracy of perhaps 60 percent. Machine learning methods might capture some of the correlations and relationships missed by human eyes, but they are themselves guesses and approximations.

Searching for identical duplicates is straightforward. Near-duplicates and those that are ambiguous create murkiness. While in this case there did not appear to be any genuine duplicates, it’s possible that a multitude of unusual errors might lead to duplicates that then distinguish themselves from one another where gender, age, marital status, and other traits are wildly different due to some of these errors. Potentially data corruption and attempted restoration? These types of incidents could not possibly be recognized by the methods used in this project (i.e. n\_column\_duplicate\_tester) and would go unnoticed, contaminating the data with seemingly authentic entries.

The IQR method is one of many and shows its limits on distributions that are heavily skewed. Ideally, detected outliers are evaluated for being potential errors or in need of special quarantining from the other data points. Being too conservative or too overzealous in the removal or modification of outliers can distort conclusions and calculated metrics. Reviewing histograms, distributions, and z-scores can assist in identifying blatant anomalies and errors, but it's imprecise and wouldn’t notice incorrect inputs that are still within the bounds set by IQR or another method. The decision to partition a dataset into separate clusters (or leave it alone) is situational and isn’t necessarily one with decisive superiority. In some cases the populations are sufficiently different that some conclusions would be more accurate if they’re treated separately while in other cases you would lose valuable information contained within the distribution’s tails. IQR bounds and histograms provide guidelines for detecting potential outliers, but do not offer any insight into determining whether they should be isolated. Nor do they verify the data entry is genuine if it’s not too anomalous.

Finding data points that violate column and reality constraints range from trivial to fiendishly difficult. Locating negative values in columns that should be positive or improper formatting is straightforward, but if there is a subtle contamination of the data entry that fails to demonstrate sufficiently noticeable anomalous behavior, it would be very difficult to determine. Even if it’s suspected, diagnosing it as genuine over a statistically significant difference worthy of further investigation is limited without additional data and research. In particular, it’s impossible to know precisely why some outage values were recorded as negative. Evidence leans towards an accidental negative sign but it also could have been a machine error that converted a number past a certain threshold into a negative one that’s not disclosed in the data dictionary. The use of population\_adjuster corrected the most egregious low population values, yet it’s also possible some of the high population values were inaccurate for other reasons, causing additional distortions in imputing the local means. Without additional follow-up and cross-validation into population values at the various cities (technically latitude and longitude coordinates), there’s no way to be certain what the accurate population values should be. Lastly, the odd behavior between income, employment status, and job would require contacting the customers to verify those values while also determining if they’re reporting their personal income or the household’s income. As there may be incorrect entries in one or more of those columns, it’s impossible to say with the current information what happened since there are multiple potential explanations. The customer’s information might have only been partially updated, entered incorrectly by a human, income estimated through some means that wasn’t mentioned in the data dictionary, or some other difficult to discern reason. Reviewing aggregate statistics helps observe the unexpected pattern but can’t provide any clarity into which values are inaccurate. It is certainly possible that the values are accurate due to unusual circumstances, such as a retired therapist who works full time but spends most of the week volunteering her services.

Overall, the limitations in treatment of nulls, outliers, duplicates, and constraint violations can impact other areas, such as anomalies affecting imputation models and introducing distortions into the nulls being overwritten.

**D. 7. Impact of limitations**

If there are undetected duplicates remaining in the cleaned data, they provide additional weight to what should be a single customer (data row), leading to inaccuracies in models and calculated statistics, which can potentially alter the conclusions drawn.

Imputed null values strive for accuracy but are naturally imperfect. An undiscovered pattern or correlation vital to accurate prediction of null values (potentially missed by humans and machine learning models) would leave the cleaned data with suboptimal imputations that inherently introduce artificial noise and errors that weren’t originally present. Depending on the analysis being performed and how many stages the noise goes through, it could potentially have a pronounced and significant impact on conclusions without anyone realizing. The uncertainty ideally is extinguished, but in some cases could unwittingly ruin additional conclusions and models.

Improperly handled outliers, dependent on the nature of the project, can pull statistics (namely the means) and machine learning models away from a more accurate representation had the outliers been excluded or quarantined.

Data points violating column constraints or constraints from reality imposed by multiple variables are either incorrect or sufficiently unusual to require further inspection. For this project in particular, if there is widespread incorrect (or at least inconsistent) entry of incomes (namely the concern that some customers report personal income while others report household income), conclusions and models drawn from income values draw from the same poisoned well and would be virtually useless. If the inaccuracies are sufficiently small or the weight of income in a model is fairly minor then the impact could be negligible, but it would be difficult to assess the impact without additional research and comparison.

Regarding the project’s research question of predicting churn, data usage, and monthly charges from the variables in this data set, the predictions would rely on machine learning and regression models. If imputed null values are significantly inaccurate, outliers are mishandled, or duplicates go unnoticed, these distortions in the data will contaminate the model’s fitting procedure and produce inaccurate parameters. In turn, those inaccurate parameters will produce less than reliable predictions.

**E. 1. Principal components**

Among the numerical columns in the dataset, 'Children', 'Email', 'Contacts', and 'Yearly\_equip\_failure' all took integer values over relatively small ranges and would have been inappropriate to use with PCA. Although 'Age' and 'Population' are integers, age is a continuous value rounded down for convenience and exists across a large enough range. Although populations should be integers, the immense range of ~10^5 makes is sufficiently close to being a continuous value that it's also included.

This left the following 9 variables for PCA: 'Age', 'Population', 'Lat', 'Lng', 'Income', 'Outage\_sec\_perweek', 'Tenure', 'MonthlyCharge', and 'Bandwidth\_GB\_Year'.

Below is the principal components loading matrix:

A screenshot of a computer

Description automatically generated

**E. 2. Criteria used**

Scree plot:

A graph with a line and a red line

Description automatically generated

Using the Kaiser rule for retaining PCA components with an eigenvalue of 1.00 or greater, there are 6 components that should be kept (variables PC1, PC2, PC3, PC4, PC5, and PC6). Reviewing the above scree plot and the explained variance ratios (see image in section E.1.), the eigenvalues plateau between five and six components, while the seventh component has an eigenvalue of ~0.87, with the eighth and ninth being even lower and less significant under the Kaiser rule. The cumulative explained variance ratios give ~82 percent explained variance by retaining six components.

**E. 3. Benefits**

Principal component analysis can assist the organization in reducing the number of dimensions (variables or data columns) used in its predictive models. This speeds up data processing and modeling while keeping the analysis focused on the most relevant and impactful variables. Additionally, it can save on storage space for organizations that deal with significant volumes of data. Highly correlated and redundant variables, as well as those that are of negligible impact, can potentially be identified in certain cases and removed from the data sampling procedures. For example, a data gathering process that initially includes as many variables as possible just in case they’re relevant can be simplified after determining their roles are so limited it would be preferable to stop collecting them. Specific to this organization, the strong correlation between tenure and annual bandwidth usage could allow some dimensionality reduction by only recording one of the values and using the linear model to estimate the other when it’s needed. This would save on data storage as well as avoiding an effective duplication of the weight of one of those variables in a model that uses both as predictor variables.

**F. Panopto video**

See attached video link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=c67fabee-9175-4edc-89ac-b14c01201b16.

**G. Sources of third-party code**

1. Donthi, Suraj. 2024. DataCamp "Dealing with Missing Data in Python". Chapter 2. Retrieved April 7, 2024, from <https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/does-missingness-have-a-pattern?ex=7>.

2. WGU. 2024. D206 Data Cleaning "Welcome to Getting Started With Principal Component Analysis". Retrieved April 7, 2024, from <https://westerngovernorsuniversity.sharepoint.com/sites/DataScienceTeam/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FDataScienceTeam%2FShared%20Documents%2FGraduate%20Team%2FD206%2FStudent%20Facing%20Resources%2FD206%20%2D%20Getting%20Started%20with%20D206%20Video%20Series%20%28Slides%20and%20Videos%29%2F7%2E%20D206%2DGettingStartedPCA%2Epdf&parent=%2Fsites%2FDataScienceTeam%2FShared%20Documents%2FGraduate%20Team%2FD206%2FStudent%20Facing%20Resources%2FD206%20%2D%20Getting%20Started%20with%20D206%20Video%20Series%20%28Slides%20and%20Videos%29>.

3. WGU. 2024. D206 Data Cleaning “Data Files and Associated Dictionary Files”. Churn Data and Dictionary Files. Retrieved April 7, 2024, from https://web5.wgu.edu/aap/content/d206-ema.html.

**H. Sources**

No additional sources were used.