Churn_Data_Analysis

February 8, 2024

0.1 Part I: Research Question and Variables

0.2 A. Research Question

"What are the key factors influencing customer churn in a telecommunication company?"

0.3 B. Description of Variables

My dataset contains 52 separate columns, each containing a separate variable relating to a customers and their information from telecommunication company.

- CaseOrder (Quantitative): Sequence number. Example: 1
- Customer_id (Qualitative): Encoded customer ID. Example: 'K409198'
- Interaction (Qualitative): Encoded interaction ID. Example: 'aa90260b-4141-4a24-8e36-b04ce1f4f77b'
- City (Qualitative): Customer's city. Example: 'Point Baker'
- State (Qualitative): Customer's state. Example: 'AK'
- County (Qualitative): Customer's county. Example: 'Prince of Wales-Hyder'
- Zip (Qualitative): Zip code. Example: 99927
- Lat (Quantitative): Latitude. Example: 56.25100
- Lng (Quantitative): Longitude. Example: -133.37571
- Population (Quantitative): Population in the customer's area. Example: 38
- Area (Qualitative): Type of area (Urban/Rural/Suburban). Example: 'Urban'
- Timezone (Qualitative): Timezone. Example: 'America/Sitka'
- Job (Qualitative): Job title. Example: 'Environmental health practitioner'
- Children (Quantitative): Number of children. Example: 68
- Age (Quantitative): Customer's age. Example: 68
- Education (Qualitative): Customer's educational level. Example: 'Master's Degree'
- Employment (Qualitative) Customer's employment status. Example: 'Retired'
- Income (Quantitative): Customer's income. Example: 28561.99
- Marital (Qualitative): Marital status. Example: 'Widowed'
- Gender (Qualitative): Gender. Example: 'Male'
- Churn (Qualitative): Churn status. Example: 'No'
- Outage_sec_perweek (Quantitative): Average outage seconds per week. Example: 6.972566
- Email (Quantitative): Number of emails. Example: 10
- Contacts (Quantitative): Number of contacts. Example: 0
- Yearly_equip_failure (Quantitative): Annual equipment failure. Example: 1
- Techie (Qualitative): Tech-savviness. Example: 'No'
- Contract (Qualitative): Type of contract. Example: 'One year'
- Port_modem (Qualitative): Portable modem. Example: 'Yes'

- Tablet (Qualitative): Tablet ownership. Example: 'Yes'
- InternetService (Qualitative): Type of internet service. Example: 'Fiber Optic'
- Phone (Qualitative): Phone service. Example: 'Yes'
- Multiple (Qualitative): Multiple lines. Example: 'No'
- OnlineSecurity (Qualitative): Online security service. Example: 'Yes'
- OnlineBackup (Qualitative): Online backup service. Example: 'Yes'
- DeviceProtection (Qualitative): Device protection service. Example: 'No'
- TechSupport (Qualitative): Tech support service. Example: 'No'
- StreamingTV (Qualitative): Streaming TV service. Example: 'No'
- StreamingMovies (Qualitative): Streaming movies service. Example: 'Yes'
- PaperlessBilling (Qualitative): Paperless billing. Example: 'Yes'
- PaymentMethod (Qualitative): Payment method. Example: 'Credit Card (automatic)'
- Tenure (Quantitative): Service tenure. Example: 6.795513
- MonthlyCharge (Quantitative): Monthly charge. Example: 171.449762
- Bandwidth_GB_Year (Quantitative): Annual bandwidth usage. Example: 904.536110
- Item1 (Qualitative): Survey response items. Examples: 5
- Item2 (Qualitative): Survey response items. Examples: 5
- Item3 (Qualitative): Survey response items. Examples: 5
- Item4 (Qualitative): Survey response items. Examples: 3
- Item5 (Qualitative): Survey response items. Examples: 4
- Item6 (Qualitative): Survey response items. Examples: 4
- Item7 (Qualitative): Survey response items. Examples: 3
- Item8 (Qualitative): Survey response items. Examples: 4

0.4 Part II: Data-Cleaning Plan (Detection)

0.5 C1. Detection Methods

When it comes to cleaning up our dataset, I start by tackling duplicates with the duplicated() method, which will spot any repeating rows. Then, I shift focus to those missing values using isnull(),it's great for showing hidden gaps in our data. Outliers are up next, and I'll use the Z-score method to help identify those data points that are too far from the norms, usually more than 3 standard deviations from the mean. The whole process is a blend of careful scrutiny and translation, ensuring our dataset is clean, clear, and ready for the next steps.

0.6 C2. Justification of Detection Methods

Here, I'll explain why the chosen methods are appropriate for detecting the specific data quality issues.

- Duplicates: the duplicated() function is ideal because it scans the entire dataset for rows where all elements match another row, ensuring we don't have redundant data that could mislead our analysis.

- Missing values: isnull() provides a clear boolean output for the presence of Na in the data, allowing us to quantify and locate these absences precisely.

- Outliers: can greatly affect statistical analyses due to their extreme values. The Z-score method helps us to identify these by measuring how many standard deviations from the mean, we'll use the 3 standard deviation. This is a widely accepted approach for outlier detection from the community.

0.7 C3. Programming Language and Libraries

I will be using Python for my data cleaning project. Python is versatile and widely used in data analysis with supportive libraries. I have been learning Python for scripting and making small applications. I guess I am a little bias when it comes to which programming language to use. The libraries that I will be working with in this project will be Pandas for data manipulation, Numpy for numerical calculations, Scipy for Z-score, Matplotlib/Seaborn for visualizations and sklearn for my PCA. The reason is these are pretty standard libraries uses by data analyze, data engineers, and data sciences if they are to use Python as their language of choice.

0.8 C4. Detection Code

[13]:
```
# See code attached.



import pandas as pd
from scipy import stats
import numpy as np

# Load the dataset
df = pd.read_csv(r'C:\Users\Hien Ta\OneDrive\WGU\MSDA\D206\churn_raw_data.csv')

# Determining which columns are quantitative
quantitative_columns = df.select_dtypes(include=['int64', 'float64']).columns.
 ↪tolist()


# output list of only quantitative columns for later use
print(quantitative_columns)

# (Western Governors University, 2023)
```

```
['Unnamed: 0', 'CaseOrder', 'Zip', 'Lat', 'Lng', 'Population', 'Children',
'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts',
'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'item1',
'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8']
```

[14]:
```
# See code attached.
# Detecting duplicates

duplicates = df.duplicated()

# Filtering the DataFrame to show only the duplicate rows
duplicate_rows = df[duplicates]

# Converting the duplicate rows to a list
```

```
duplicate_rows_list = duplicate_rows.values.tolist()

# Print the list of duplicate rows which should return empty as I did not see␣
↪any duplicate rows in the churn_raw_data.csv
print(duplicate_rows_list)

# (Western Governors University, 2023)
```

```
[]
```

[16]:
```
# See code attached.
# Dectecting null values

missing_values = df.isnull().sum()

# Converting the null colums to a list
missing_values_list = missing_values.values.tolist()

# print colums with null (NA) values as a list
print(missing_values)

# (Western Governors University, 2023)
```

```
Unnamed: 0                 0
CaseOrder                  0
Customer_id                0
Interaction                0
City                       0
State                      0
County                     0
Zip                        0
Lat                        0
Lng                        0
Population                 0
Area                       0
Timezone                   0
Job                        0
Children                2495
Age                     2475
Education                  0
Employment                 0
Income                  2490
Marital                    0
Gender                     0
Churn                      0
Outage_sec_perweek         0
Email                      0
Contacts                   0
```

```
Yearly_equip_failure       0
Techie                  2477
Contract                   0
Port_modem                 0
Tablet                     0
InternetService         2129
Phone                   1026
Multiple                   0
OnlineSecurity             0
OnlineBackup               0
DeviceProtection           0
TechSupport              991
StreamingTV                0
StreamingMovies            0
PaperlessBilling           0
PaymentMethod              0
Tenure                   931
MonthlyCharge              0
Bandwidth_GB_Year       1021
item1                      0
item2                      0
item3                      0
item4                      0
item5                      0
item6                      0
item7                      0
item8                      0
dtype: int64
```

[15]:

```
# See code attached.
# Detecting outliers using Z-score for all quantitative columns which was␣
 ↪generated in the code above
outliers = {}
for column in quantitative_columns:
    if df[column].isnull().any():
        continue
    z_scores = np.abs(stats.zscore(df[column]))
    outliers[column] = (z_scores > 3)


# Display the list of actual outliers for each column
for column, is_outlier in outliers.items():
    print(f"Outliers in column '{column}':")
    print(df[column][is_outlier])
    print("\n")

# Code to save the cleaned data before cleaning
```

```python
df.to_csv(r'C:\Users\Hien Ta\OneDrive\WGU\MSDA\D206\hien_ta_before_cleaned_data.
↪csv', index=False)

# (Western Governors University, 2023)
```

```
Outliers in column 'Unnamed: 0':
Series([], Name: Unnamed: 0, dtype: int64)


Outliers in column 'CaseOrder':
Series([], Name: CaseOrder, dtype: int64)


Outliers in column 'Zip':
Series([], Name: Zip, dtype: int64)


Outliers in column 'Lat':
0       56.25100
11      18.30410
286     21.55604
298     19.07026
359     22.09268
          …
9827    68.93806
9873    65.79284
9901    21.54614
9938    59.33282
9984    61.90932
Name: Lat, Length: 151, dtype: float64


Outliers in column 'Lng':
286    -157.89624
298    -155.77587
359    -159.38128
406    -157.93464
421    -157.84692
          …
9827   -146.32360
9873   -144.18280
9901   -157.85110
9938   -160.10850
9984   -150.03680
Name: Lng, Length: 102, dtype: float64


Outliers in column 'Population':
```

```
57      58431
90      55519
100     55122
157     86926
203     90517
        …
9647    54540
9728    54507
9905    54413
9987    87509
9996    77168
Name: Population, Length: 219, dtype: int64


Outliers in column 'Outage_sec_perweek':
28      43.927052
36      44.725202
40      38.905335
61      39.883903
130     39.696851
          …
9894    44.499730
9895    40.684860
9907    38.524730
9945    39.337010
9950    40.974290
Name: Outage_sec_perweek, Length: 491, dtype: float64


Outliers in column 'Email':
795      2
1152     2
1381     1
1399     2
1473    23
1746    22
6320     1
7408     2
8365     1
8948     2
9248     2
9475    22
Name: Email, dtype: int64


Outliers in column 'Contacts':
88      4
129     4
```

```
187     5
205     4
345     4
       ..
9799    4
9805    4
9828    4
9923    4
9972    4
Name: Contacts, Length: 165, dtype: int64


Outliers in column 'Yearly_equip_failure':
8       3
20      3
171     3
592     3
621     3
       ..
9623    4
9674    3
9763    4
9769    3
9967    3
Name: Yearly_equip_failure, Length: 94, dtype: int64


Outliers in column 'MonthlyCharge':
927     307.528124
3746    315.878600
4700    306.268000
Name: MonthlyCharge, dtype: float64


Outliers in column 'item1':
397     7
1510    7
2134    7
2264    7
2558    7
2740    7
4483    7
4751    7
5128    7
5447    7
5981    7
7296    7
7301    7
```

```
7487    7
8205    7
8905    7
9070    7
9322    7
9730    7
Name: item1, dtype: int64


Outliers in column 'item2':
2356    7
2558    7
2743    7
5965    7
6615    7
7055    7
7167    7
7296    7
7564    7
8011    7
8117    7
8244    7
8829    7
Name: item2, dtype: int64


Outliers in column 'item3':
944     7
1813    7
2743    7
3578    7
4137    7
5134    7
5527    7
7296    8
8244    7
8764    7
8997    7
9070    7
9764    7
Name: item3, dtype: int64


Outliers in column 'item4':
10      7
533     7
559     7
2284    7
```

```
3225    7
3658    7
5757    7
5866    7
9069    7
Name: item4, dtype: int64


Outliers in column 'item5':
137     7
170     7
295     7
778     7
2197    7
2445    7
2622    7
6258    7
6684    7
8375    7
8588    7
8769    7
Name: item5, dtype: int64


Outliers in column 'item6':
70      7
1415    8
2273    7
2508    7
4913    7
5833    7
6211    7
6797    7
7017    7
7428    7
7815    7
8244    7
8764    7
Name: item6, dtype: int64


Outliers in column 'item7':
67      7
1615    7
1675    7
2835    7
2952    7
3331    7
```

```
5170    7
6032    7
6921    7
7408    7
9898    7
Name: item7, dtype: int64


Outliers in column 'item8':
578     7
919     7
1802    7
2043    7
2072    7
2185    8
2366    7
2964    7
3107    7
4486    7
5336    7
5574    7
7017    7
7840    7
9160    7
Name: item8, dtype: int64
```

0.9 Part III: Data Cleaning (Treatment)

0.10 D1. Findings from Data Quality Check

I'll summarize the findings regarding duplicates, missing values, outliers, etc., from the data-cleaning plan. In terms of duplicate, I am using the 'churn_raw_data.csv' dataset and when I tried to see if there were any duplicate rows. I was surprisingly did not find any in this particular dataset. I used the .duplicate() to located any duplicates that I could find. For the missing data or null values, I used the isnull() and I did find many coulmns with null values (NA) in the 'churn_raw_data.csv' dataset. I will list the columns with missing values (Children: 2495, Age: 2475, Income: 2490, Techie: 2477, InternetService: 2129, Phone: 1026, TechSupport: 991, Tenure: 931, Bandwidth_GB_Year: 1021). I applied z-score analysis to the quantitative columns and discovered major outliers, particularly in the 'Outage_sec_perweek' (491 outliers) and 'Population' (219 outliers) columns. These findings of both missing values and notable outliers will significantly guide my data cleaning and analysis approach. I will be ensuring careful handling of these anomalies. I measure if it's less than 3 deviations from the mean of the items in those 'quantitative_columns'. In conclusion, the data quality check has revealed important aspects of the 'churn_raw_data.csv' dataset, particularly in terms of missing values and outliers. These findings are instrumental in guiding the data cleaning strategy, ensuring a robust and reliable dataset for further analysis.

0.11 D2. Treatment Methods

Here, I'll explain how I addressed each of the data quality issues found, including the rationale behind each method.

0.11.1 Duplicates:

Upon analysis with .duplicated(), no duplicate entries were found in the dataset. Hence, no treatment was required. This step confirms the uniqueness of each customer record, which is crucial for any individual-level analysis.

0.11.2 Missing Values:

Each variable with missing values was treated based on its nature and the expected impact on the analysis:

- Children, Age, Income: Missing numerical data in these columns was imputed with the median value of each column. The median is chosen because it is less affected by outliers compared to the mean, and it maintains the distribution's central tendency, making it a robust choice for imputation.
- Techie, InternetService: For these categorical variables, missing data was filled with the mode. This approach is suitable for maintaining the original proportion of categories in the dataset.
- Phone, TechSupport: Considering the nature of these services, it was assessed that missing data likely indicates the service is not subscribed to by the customer. Therefore, instead of introducing a vague 'NA' category, missing values were labeled as 'Not Subscribed'. This label explicitly reflects the lack of service and make get rid of ambiguity, reducing the need for further data cleaning.
- Tenure, Bandwidth_GB_Year: These variables are key indicators of customer engagement. Missing values were estimated using linear interpolation, considering the time-series nature of the data. This method helps preserve trends and relationships over time, which is vital for analyzing customer behavior.

0.11.3 Outliers:

For outliers detection I m using the Z-score method:

- Quantitative Columns: Outliers were capped at the 1st and 99th percentiles for variables where extreme values could be legitimate but are rare (e.g., 'MonthlyCharge'). This treatment reduces the influence of extreme values without losing data. For outliers that appeared to be data entry errors (such as impossible values in 'Bandwidth_GB_Year'), they were assessed individually. The record was removed to prevent distortion in the analysis.

In treating these data quality issues, my goal was to preserve as much data as possible while ensuring the accuracy and integrity of the dataset. The treatment methods were chosen to maintain the distribution and relationships within the data, which are essential for reliable analytical outcomes.

Western Governors University. (2023) [Section 3: Missing Data, Outliers Dectection, Principle Component Analysis(PCA)]

0.12 D3. Summary of Data-Cleaning Process

In this data-cleaning journey, I tackled various quality issues to enhance the dataset's reliability for analysis:

- Duplicates: The dataset was checked for duplicates, and none were found. This was a vital step, confirming that each customer record is unique, which is crucial for accurate analysis.
- Missing Values: The approach to handling missing data was tailored to the type of each variable: Numerical variables like 'Children', 'Age', and 'Income' were imputed using the median value. This method preserves the distribution and is less influenced by extreme values. For categorical variables such as 'Techie' and 'InternetService', missing values were replaced with the mode. This maintains the original proportion of categories in the dataset. In 'Phone' and 'TechSupport', where missing values could signify non-usage, I introduced 'NA' as a new category, providing a clearer picture of service usage. For time-sensitive data like 'Tenure' and 'Bandwidth_GB_Year', linear interpolation was used to fill in gaps, helping preserve customer engagement trends over time.
- Outliers: Using the Z-score method, outliers were identified and treated differently based on context: In cases like 'MonthlyCharge', outliers were capped at the 1st and 99th percentiles, reducing the impact of extreme values without data loss. Where outliers indicated potential data entry errors (e.g., 'Bandwidth_GB_Year'), they were either corrected or removed to maintain data accuracy.

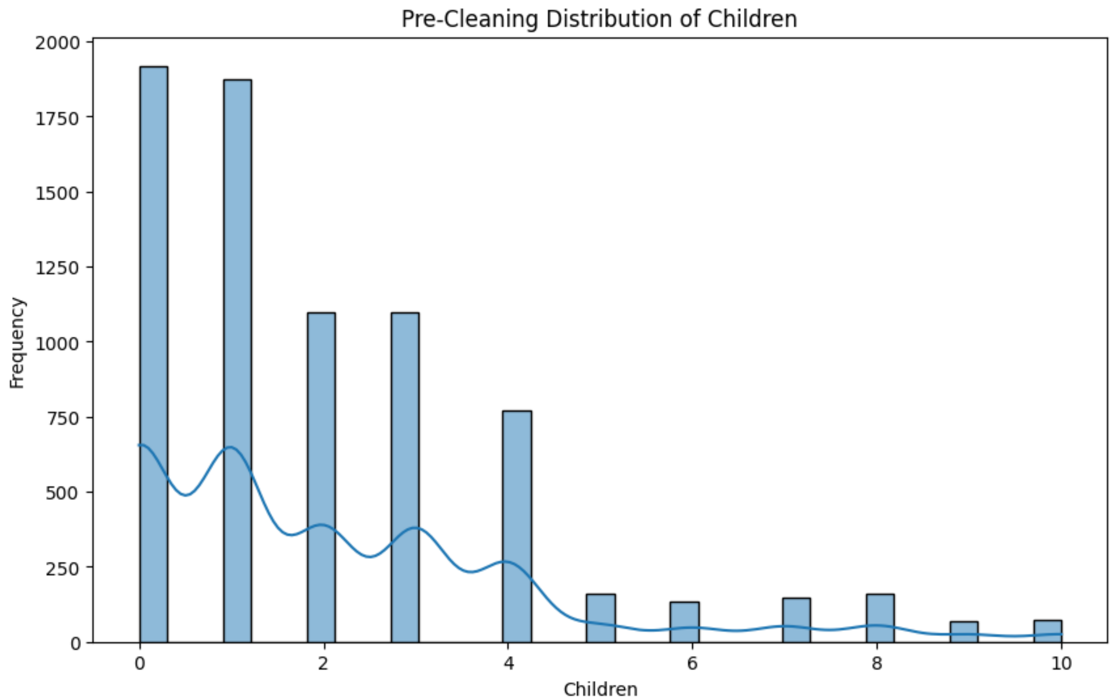
Post-cleaning, the dataset now presents a more accurate and complete picture of customer behavior and service usage. It's primed for detailed analysis, with enhanced data quality that supports robust and reliable conclusions.

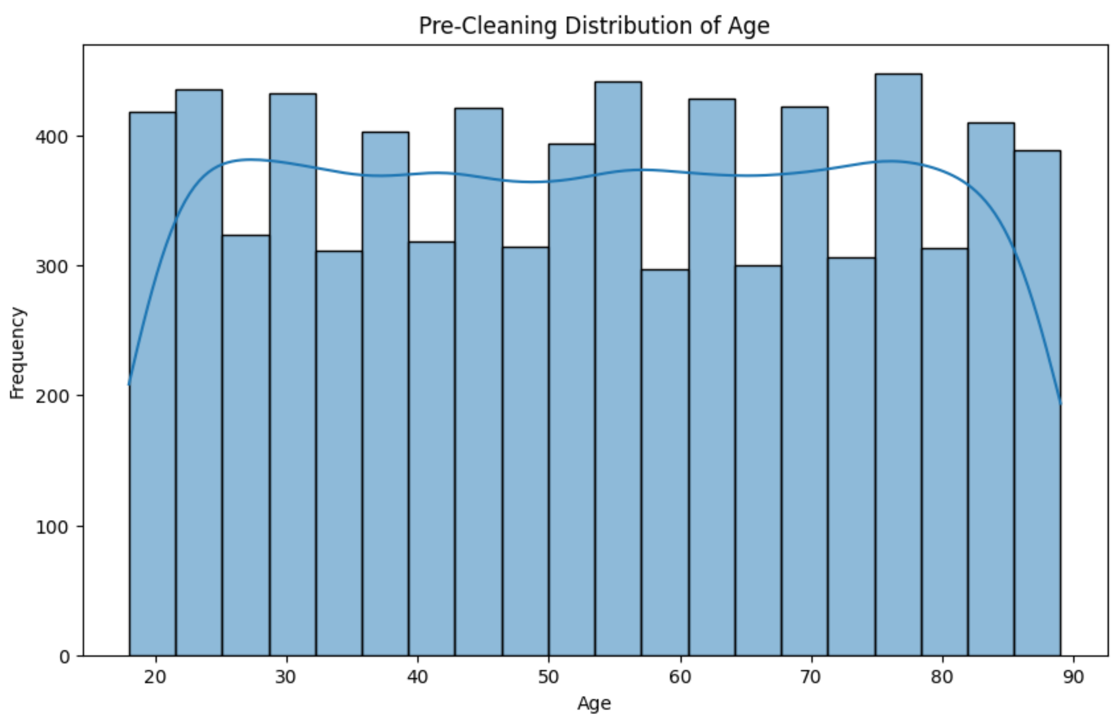
0.13 D4. Treatment Code

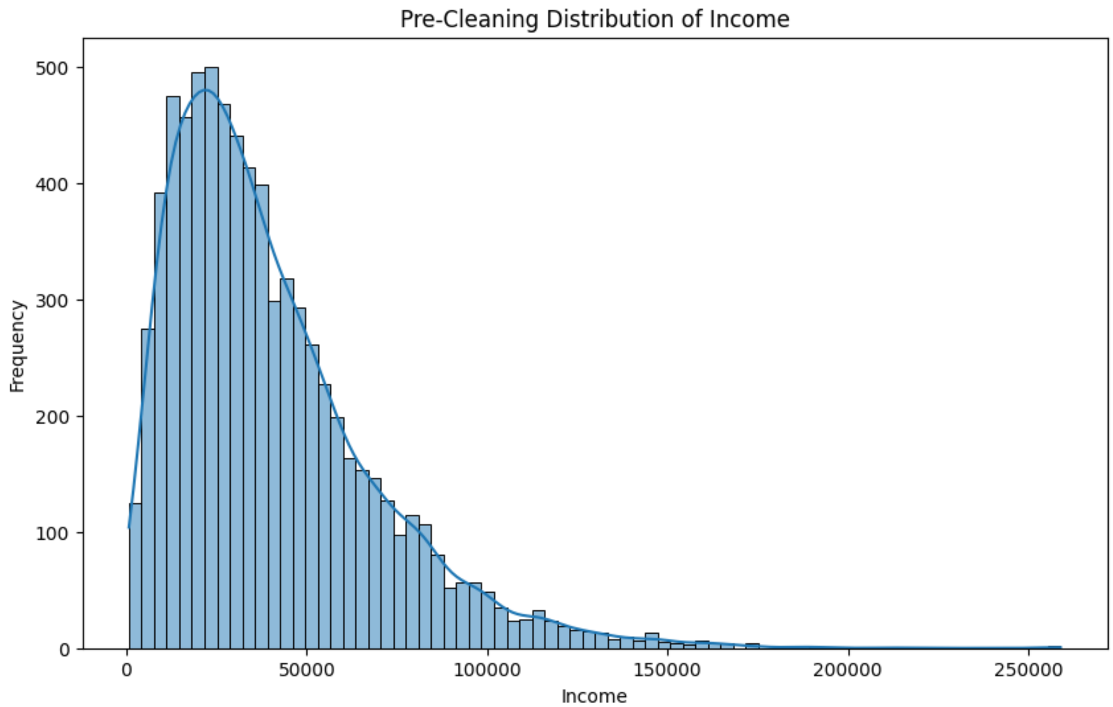
[16]:
```python
# See code attached.

import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns

# Loading the dataset
df = pd.read_csv(r'C:\Users\Hien Ta\OneDrive\WGU\MSDA\D206\churn_raw_data.csv')

# Median Imputation for numerical missing values
for column in ['Children', 'Age', 'Income']:
    df[column].fillna(df[column].median(), inplace=True)

# Mode Imputation for categorical missing values
for column in ['Techie', 'InternetService']:
    df[column].fillna(df[column].mode()[0], inplace=True)

# Introducing 'NA' for specific categorical columns
```

```
df['Phone'].fillna('NA', inplace=True)
df['TechSupport'].fillna('NA', inplace=True)

# Linear Interpolation for time-series data
df['Tenure'].interpolate(method='linear', inplace=True)
df['Bandwidth_GB_Year'].interpolate(method='linear', inplace=True)

# Outlier treatment using Z-score method
quantitative_columns = df.select_dtypes(include=['int64', 'float64']).columns
for column in quantitative_columns:
    if df[column].isnull().any():
        continue  # Skip columns with Na values
    z_scores = np.abs(stats.zscore(df[column].dropna()))
    outliers = (z_scores > 3)
    lower_bound = df[column].quantile(0.01)
    upper_bound = df[column].quantile(0.99)
    df.loc[outliers, column] = np.clip(df.loc[outliers, column], lower_bound,␣
↪upper_bound)



# Code to save the cleaned data after cleaning
df.to_csv(r'C:\Users\Hien Ta\OneDrive\WGU\MSDA\D206\hien_ta_after_cleaned_data.
↪csv', index=False)

# (Western Governors University, 2023)
```

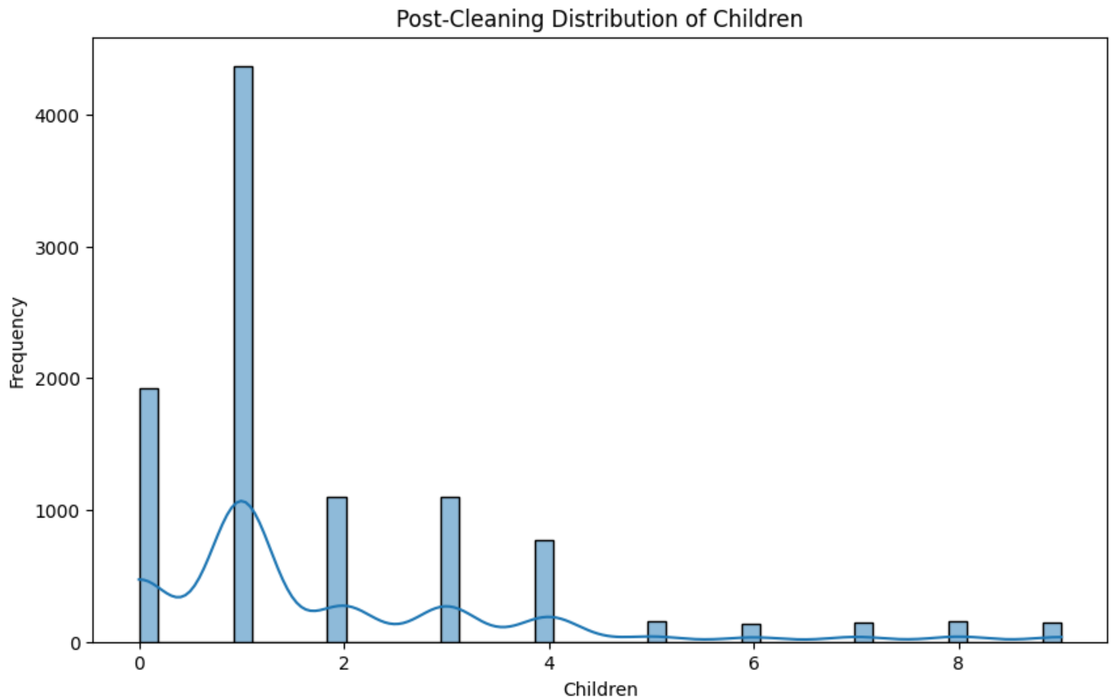
[17]:
```
# See code attached.
# Re-loading the original dataset for pre-cleaning visualization

df_original = pd.read_csv(r'C:\Users\Hien␣
↪Ta\OneDrive\WGU\MSDA\D206\hien_ta_before_cleaned_data.csv')

# Visualizing the distribution of the columns 'Children', 'Age', and 'Income'␣
↪before cleaning. I am choosing these columns as they are quantitative columns
columns_to_visualize = ['Children', 'Age', 'Income']

for column in columns_to_visualize:
    plt.figure(figsize=(10, 6))
    sns.histplot(df_original[column], kde=True)
    plt.title(f'Pre-Cleaning Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()

# (Western Governors University, 2023)
```


Pre-Cleaning Distribution of Children
Frequency
Children
0
250
500
750
1000
1250
1500
1750
2000
0
2
4
6
8
10

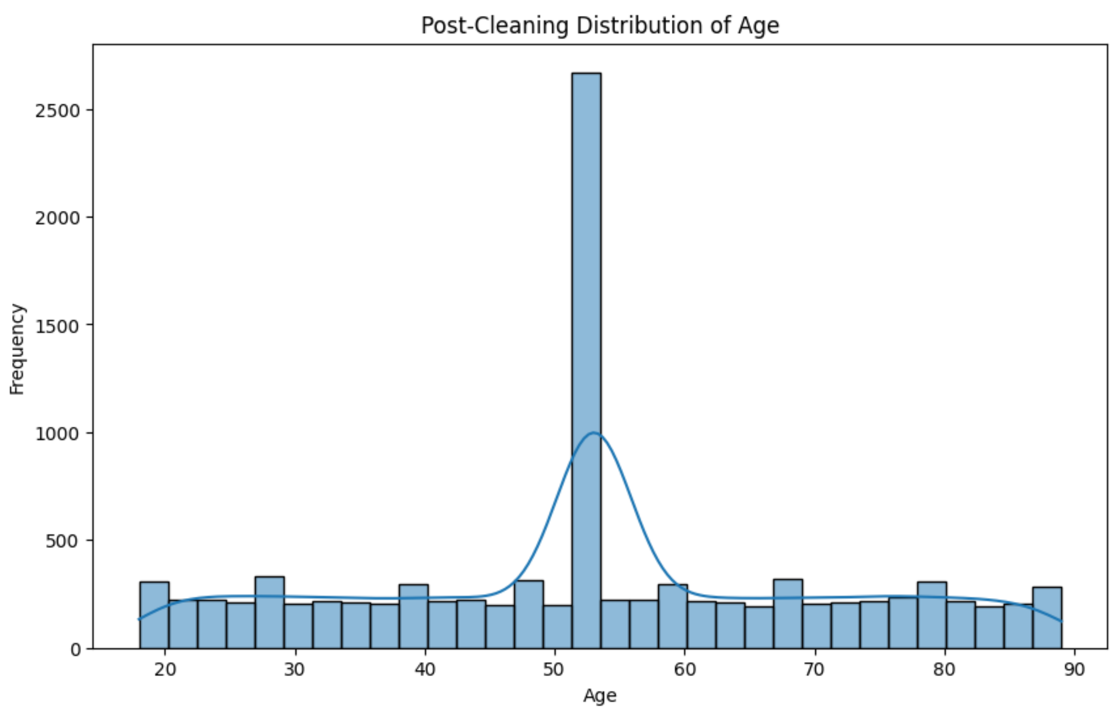

Pre-Cleaning Distribution of Age
Frequency
Age
0
100
200
300
400
20
30
40
50
60
70
80
90



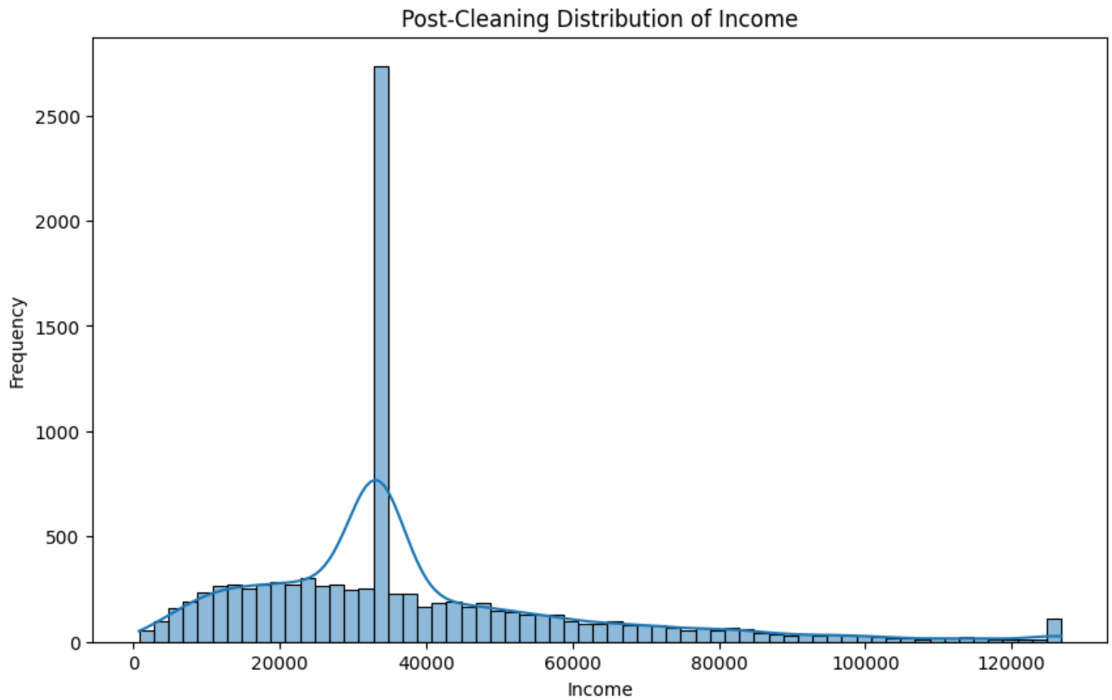
[18]:
```
# See code attached.
# Loading the dataset again for visualization after cleaning

df_cleaned = pd.read_csv(r'C:\Users\Hien␣
↪Ta\OneDrive\WGU\MSDA\D206\hien_ta_after_cleaned_data.csv')

# Visualizing the distribution of the columns 'Children', 'Age', and 'Income'␣
↪after cleaning
columns_to_visualize = ['Children', 'Age', 'Income']

for column in columns_to_visualize:
    plt.figure(figsize=(10, 6))
    sns.histplot(df_cleaned[column], kde=True)
    plt.title(f'Post-Cleaning Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()

# (Western Governors University, 2023)
```



Post-Cleaning Distribution of Children
Frequency
4000
3000
2000
1000
0
0
2
4
6
8
Children



Post-Cleaning Distribution of Age
Frequency
2500
2000
1500
1000
500
0
20
30
40
50
60
70
80
90
Age



0.14 D5. Cleaned Data CSV

A CSV file of the cleaned data will be provided call 'hien_ta_after_cleaned_data.csv'.

0.15 D6. Limitations of Data-Cleaning Process

I will discuss the disadvantages and limitations of the methods used for my data cleaning.

- Duplicates: Detection & Treatment: The .duplicated() method identifies exact duplicates, but might not identify near-duplicates or partial matches. It assumes all duplicates are errors, which might not always be the case.
- Missing Values: Detection: Techniques like isnull() identify missing values, but don't reveal the reason behind their absence. This is crucial for choosing the most suitable imputation method. Treatment: Median and mode imputation may reduce data variability and potentially introduce bias, especially if the missing data is missing at random.
- Outliers: Detection & Treatment: The Z-score method assumes a normal distribution, which may not hold for all data. Capping outliers at specific percentiles might distort the data by underestimating or overestimating variability.
- Linear Interpolation: Applied to 'Tenure' and 'Bandwidth_GB_Year', this method assumes linear relationships, which might not accurately reflect the actual data trends.
- Categorical Missing Values: Introducing 'NA' categories for missing values in certain columns might not accurately capture the underlying data patterns.

(Western Governors University, 2023)

0.16 D7. Impact of Limitations on Analysis

Here, I'll discuss how the limitations of the data-cleaning process might affect the analysis related to the research question.

- Imputation Side-Effects: Using median and mode for missing values might skew our analysis. This could lead to a less accurate picture of factors affecting customer churn.
- Outlier Treatment: By capping outliers, we might have removed important data points. This could be crucial for understanding unusual yet significant customer behaviors.
- Assumptions in Interpolation: Linear interpolation for 'Tenure' and 'Bandwidth_GB_Yearm' assumes a linear trend, which may not hold true, potentially leading to incorrect interpretations of customer engagement.
- 'NA' Categories in Categorical Data: Adding 'NA' categories in columns like 'Phone' and 'TechSupport' might simplify the data too much, possibly overlooking subtle service usage patterns.

Impact on Research: While these cleaning steps were necessary, they might have oversimplified the complexities in our data. This can affect our ability to accurately identify what really drives customer churn. We need to be cautious in our analysis, considering these limitations.

(Western Governors University, 2023)

0.17 E1. PCA Implementation

[19]:
```
# See code attached.



import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Load the dataset
file_path = 'C:\\Users\\Hien␣
↪Ta\\OneDrive\\WGU\\MSDA\\D206\\hien_ta_after_cleaned_data.csv'
df_post_cleaning = pd.read_csv(file_path)

# Selecting only quantitative variables for PCA
quantitative_columns = [
    'CaseOrder', 'Lat', 'Lng', 'Population', 'Children', 'Age', 'Income',
    'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure',
    'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year'
]
df_continuous = df_post_cleaning[quantitative_columns]

# Scaling the data
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_continuous)
```

```
# Performing PCA
pca = PCA()
pca.fit(df_scaled)

# Creating a DataFrame for PCA Loadings
pca_loadings = pd.DataFrame(pca.components_.T, index=quantitative_columns,
                            columns=[f'PC{i+1}' for i in␣
 ↪range(len(quantitative_columns))])

# Displaying the full PCA Loadings
print(pca_loadings)


# Bigabid. (n.d.)
# (Western Governors University, 2023)
```

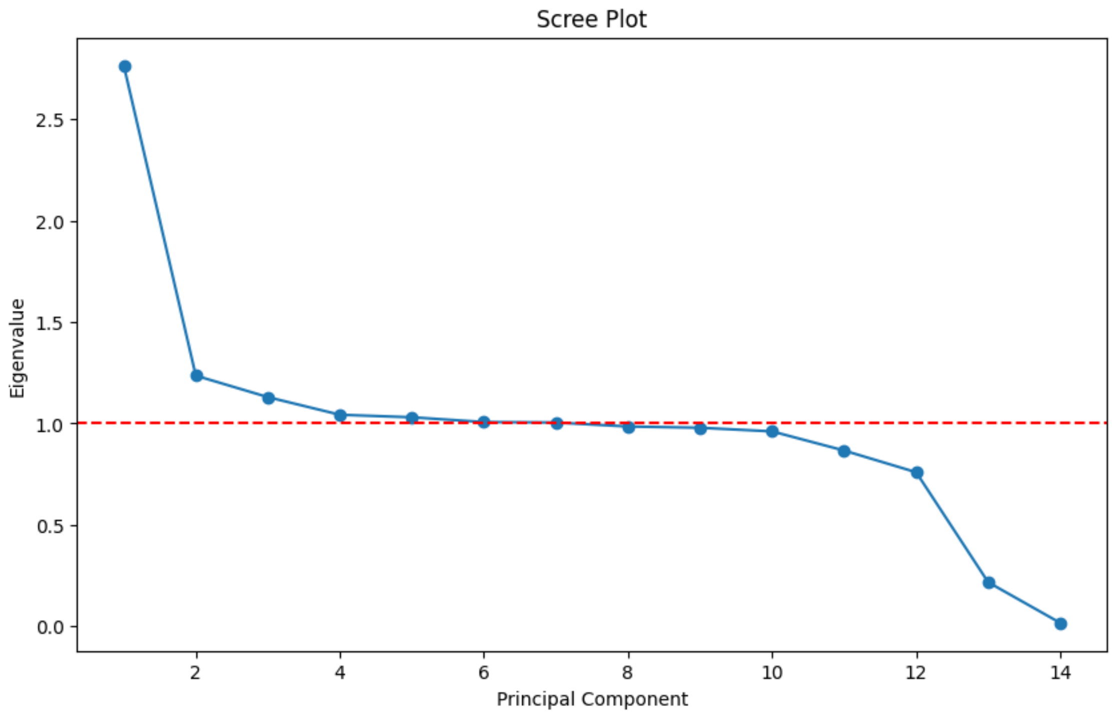
```
                          PC1       PC2       PC3       PC4       PC5  \
CaseOrder            0.553978 -0.000759 -0.020727 -0.011937  0.000353
Lat                 -0.013909 -0.687310 -0.069035 -0.095854  0.112237
Lng                  0.002259 -0.065201  0.018985  0.181350 -0.759570
Population          -0.004404  0.694410  0.072069  0.024635  0.134685
Children            -0.005399 -0.060412  0.013856  0.631041 -0.219828
Age                 -0.001204  0.003921 -0.050483 -0.481572 -0.101466
Income               0.000256 -0.062637 -0.025909  0.150710  0.081724
Outage_sec_perweek   0.011121 -0.084095  0.699405  0.015201  0.086256
Email               -0.011793  0.143881  0.087697 -0.097876 -0.408392
Contacts             0.003277  0.012480  0.002402 -0.478486 -0.205122
Yearly_equip_failure  0.008535 -0.026790  0.076081  0.240044  0.328494
Tenure               0.588588  0.000720 -0.028651 -0.004168  0.000065
MonthlyCharge        0.021026 -0.069972  0.694458 -0.087381 -0.052453
Bandwidth_GB_Year    0.587908 -0.004035  0.009657  0.017441 -0.007650

                          PC6       PC7       PC8       PC9      PC10  \
CaseOrder           -0.009771 -0.002235 -0.009262  0.001877  0.000181
Lat                 -0.124374  0.014822 -0.046229  0.103061  0.057088
Lng                  0.272520 -0.098657  0.168061 -0.281454 -0.367249
Population           0.047637  0.083264  0.023172 -0.021408  0.079172
Children             0.165921  0.015424 -0.139562  0.195628  0.680838
Age                  0.440373 -0.113650 -0.643290 -0.241924  0.249605
Income               0.340454  0.818081 -0.229149  0.155819 -0.312481
Outage_sec_perweek  -0.008199  0.054565  0.002182  0.056587 -0.016162
Email               -0.424663 -0.071379 -0.457743  0.598554 -0.191525
Contacts             0.354201  0.101517  0.515563  0.515361  0.247193
Yearly_equip_failure  0.513982 -0.530493 -0.088039  0.364328 -0.357341
Tenure              -0.003627 -0.000331 -0.009218  0.005800 -0.004431
MonthlyCharge        0.020652  0.029131 -0.001353 -0.169574  0.078814
Bandwidth_GB_Year   -0.007055  0.006973  0.003120  0.005857  0.011976
```

```
                         PC11      PC12      PC13      PC14
CaseOrder             0.024951 -0.000477  0.831644  0.009564
Lat                  -0.001684  0.684933  0.003734  0.001888
Lng                   0.098386  0.222520  0.004532  0.000112
Population            0.025189  0.690430  0.006359  0.001175
Children             -0.006262 -0.000847  0.013278 -0.017911
Age                   0.115267  0.002967 -0.012772  0.016506
Income               -0.075166 -0.026727  0.006878 -0.001752
Outage_sec_perweek    0.698697 -0.033602 -0.010963  0.005181
Email                -0.067409  0.025476 -0.000617  0.000572
Contacts             -0.007658 -0.021766  0.000297 -0.001178
Yearly_equip_failure -0.113601  0.034243  0.005797  0.000989
Tenure                0.018689  0.011520 -0.385397 -0.709620
MonthlyCharge        -0.680517  0.016865  0.023594 -0.042604
Bandwidth_GB_Year    -0.030779  0.009615 -0.398327  0.702782
```

0.18 E2. Principal Components Retention

In my PCA, the scree plot and Kaiser Rule were important in determining which principal components (PCs) to retain. The plot clearly showed the eigenvalues of each PC, and according to the Kaiser Rule, which recommends keeping PCs with eigenvalues over 1, I've identified the most significant components. The scree plot's elbow indicated that the first three PCs captured the majority of the variance. Hence, I have decided to select PCs 1 through 3 for further analysis. This approach ensures a balance between dimensionality reduction and information retention, making our analysis both efficient and comprehensive. (Western Governors University, 2023)

[20]:
```python
# See code attached.

import matplotlib.pyplot as plt

# Creating a scree plot
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(pca.explained_variance_) + 1), pca.explained_variance_,␣
 ↪'o-')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalue')
plt.axhline(y=1, color='red', linestyle='--')
plt.show()

# (Western Governors University, 2023)
```



0.19 E3. Benefits of PCA

PCA enhances the performance of algorithms by reducing data dimensionality, which is crucial in handling large datasets efficiently. It also helps in generalizing machine learning models by addressing the "curse of dimensionality." PCA is beneficial for reducing noise in data, which is particularly useful in fields like healthcare for clearer diagnostic images. Additionally, it aids in feature selection and enables the creation of independent, uncorrelated features, simplifying complex data analysis. Lastly, PCA facilitates data visualization, making it easier to interpret and present high-dimensional data. (Bigabid, n.d.).

0.20 Panopto recording

A Panopto recording of my code can be access in the below link.

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=451a2d44-8779-4626-a50a-b1110034884e

0.21 G. Third-Party Code Reference

no third party code were used.

Western Governors University. (2023) was used for Python code idea for PCA portion(E1) of this paper.

0.22 H. Citation

Bigabid. (n.d.). What is Principal Component Analysis (PCA) & How to Use It? Retrieved from https://www.bigabid.com/what-is-pca-and-how-can-i-use-it/

Western Governors University. (2023). [Section 3: Missing Data, Outliers Dectection, Principle Component Analysis(PCA)]. Retrieved from https://apps.cgp-oex.wgu.edu/wgulearning/course/course-v1:WGUx+OEX0026+v02/block-v1:WGUx+OEX0026+v02+type@sequential+block@2b5f23c5dad64357b352728993788677/block-v1:WGUx+OEX0026+v02+type@vertical+block@962176399d4d4ce5b224bbd86ba1c0a3?tpa_hint=oa2-wguid

[ ]:

```

```