**Performance Assessment: NUM3 — NUM3 Task 1: Data Cleaning**

Bader Ale

Department of Information Technology, Western Governors University

D206 Data Cleaning

Professor Eric Straw

March 31, 2023

**Table of Contents**

[Part I: Research Question and Variables 3](#_Toc133350880)

[A: Research Question: 3](#_Toc133350881)

[B: Variable List and Datatypes: 3](#_Toc133350882)

[Part II: Data-Cleaning Plan (Detection) 5](#_Toc133350883)

[C1. Methods/Functions Used 5](#_Toc133350884)

[C2. Reasoning 6](#_Toc133350885)

[C3. Programming Language 7](#_Toc133350886)

[C4. Code 7](#_Toc133350887)

[Part III: Data Cleaning (Treatment) 7](#_Toc133350888)

[D1. Detection Results 7](#_Toc133350889)

[D2. Treatment 8](#_Toc133350890)

[D3. Summarized Work 10](#_Toc133350891)

[Table of Figures 12](#_Toc133350892)

[Works Cited 13](#_Toc133350893)

# Part I: Research Question and Variables

## A: Research Question:

For this performance assessment, my research question is:Is there a relation between the number of times the primary physician visited the patient during their hospital stay and the occurrence of readmission within 30 days following the patient's discharge from the facility?

## B: Variable List and Datatypes:

The following table contains all variables contained in the original dataset along with their respective datatypes and an example.

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Name** | **Datatype** | **Description** | **Example** |
| Unnamed: 0 | int64 | An integer used as an index in original dataframe | 0, 1, 2 |
| CaseOrder | int64 | Variable used to define order of cases | 0, 1, 2 |
| Customer\_id | object | An ID that defines a specific patient | C412403 |
| Interaction | object | Internal ID for patient and corresponding procedures | 8cd49b13-f45a-4b47-a2bd-173ffa932c2f |
| UID | object | Internal ID for patient and corresponding procedures | 3a83ddb66e2ae73798bdf1d705dc0932 |
| City | object | Patient's city of residence | Eva |
| State | object | Patient's state of residence in two letter code | AL (Alabama) |
| County | object | Patient's county of residence | Morgan |
| Zip | int64 | Patient's zipcode | 35621 |
| Lat | float64 | Latitude of patient's billing address | 34.3496 |
| Lng | float64 | Longitude of patient's billing address | -86.72508 |
| Population | int64 | Population within 1 mile radius of patient | 2951 |
| Area | object | Rural, Suburban or Urban | Suburban |
| Timezone | object | Timezone of patient residence | America/Chicago |
| Job | object | Job of patient or insurance holder | Psychologist, sport and exercise |
| Children | float64 | Number of children in patient's househol | 1.0 |
| Age | float64 | Age of patient | 53.0 |
| Education | object | Highest degree earned by patient | Some College, Less than 1 Year |
| Employment | object | Employment status of patient | Full Time |
| Income | float64 | Annual income of patient | 86575.93 |
| Marital | object | Marital Status of patient | Divorced |
| Gender | object | Patient Self-Identification (Male, Female, Non-Binary | Male |
| ReAdmis | object | If patient was readmitted within 30 days of discharge | No |
| VitD\_levels | float64 | Patient Vitamin D levels (ng/mL) | 17.80233 |
| Doc\_visits | int64 | Number of times patient was visited in hospital by doctor | 6 |
| Full\_meals\_eat | int64 | Number of full meals eaten by patient in hospital (0=partial meals) | 0 |
| VitD\_supp | int64 | Number of times Vitamin D supplements were given to patient | 0 |
| Soft\_drink | object | Whether patient drinks 3 or more soft drinks in a day | Yes |
| Initial\_admin | object | How the patient was admitted to hospital (emergency, elective, observation) | Emergency Admission |
| HighBlood | object | Whether patient has high blood pressure | Yes |
| Stroke | object | Whether patient has had a stroke | No |
| Complication\_r | object | Level of patient's acuity assessed by doctor | Medium |
| Overweight | float64 | Whether patient is considered overweight (Yes=1, No=0) | 0.0 |
| Arthritis | object | Whether patient has arthritis | Yes |
| Diabetes | object | Whether patient has diabetes | Yes |
| Hyperlipidemia | object | Whether patient has hyperlipidemia | No |
| BackPain | object | Whether patient has chronic back pain | Yes |
| Anxiety | float64 | Whether patient has chronic anxiety (Yes=1, No=0) | 1.0 |
| Allergic\_rhini | object | Whether patient has allergic rhinitis | Yes |
| Reflux\_esophag | object | Whether patient has acid reflux | No |
| Asthma | object | Whether patient has asthma | Yes |
| Services | object | What service patient received while hospitalized | Blood Work |
| Initial\_days | float64 | Length of Stay of patient | 10.58577 |
| TotalCharge | float64 | Average daily amount billed to patient | 3191.04877 |
| Additional\_cha | float64 | Average daily amount billed to patient for additional procedures | 17939.40342 |
| Item1 | int64 | Survey Answer: Timely Admission (1 = most important,8 = least important) | 3 |
| Item2 | int64 | Survey Answer: Timely Treatment(1 = most important, 8 = least important) | 3 |
| Item3 | int64 | Survey Answer: Timely Visit (1 = most important, 8 = least important) | 2 |
| Item4 | int64 | Survey Answer: Reliability (1 = most important, 8 = least important) | 2 |
| Item5 | int64 | Survey Answer: Options (1 = most important, 8 = least important) | 4 |
| Item6 | int64 | Survey Answer: Hours of Treatment (1 = most important, 8 = least important) | 3 |
| Item7 | int64 | Survey Answer: Courteous Staff (1 = most important, 8 = least important) | 3 |
| Item8 | int64 | Survey Answer: Evidence of active listening from doctor (1 = most important, 8 = least important) | 4 |

# Part II: Data-Cleaning Plan (Detection)

## C1. Methods/Functions Used

In this section, we will discuss the methods (functions) were used to detect for duplicates, missing values and outliers. We started by importing the pandas python library into our notebook. The method that was used to detect duplicates was the .*duplicated*  and *.value\_counts()* method; the *.duplicated* method allows for the detection of duplicated values in a dataframe while *.value\_counts* allowed for a counts of these duplicated values, if any (NumFOCUS, Inc, 2023). As shown in the code snippet below, no duplicates were found. This was done by comparing the shape of the dataframe by running the *.shape* command with the output of *.value\_counts.* Moreover, the *.duplicated* method itself would return a Boolean value if any duplicated values were to exist.

Text

Description automatically generated with medium confidence

Figure 1: Shape of Original Dataframe

Graphical user interface, text, application

Description automatically generated

Figure 2: Duplicated Method

In order to detect missing values in the data frame, the .*isnull().sum()* command was used. This command involved detecting if any missing values existed and if so, total them by variable.

A picture containing logo

Description automatically generated

Figure 3: Code Snippet for Detecting Missing Values

From this, the variables **Children**, **Age**, **Income**, **Soft\_drink**, **Overweight**, **Anxiety** and **Initial\_days** were found to have missing values. Children, Age, Income and Initial days were quantitative variables (there were numerical in nature) while Overweight, Soft\_drink and Anxiety were qualitative as they were of the Yes/No kind although Soft\_drink was not re-expressed meaning the records were either yes or no. To solve this, ordinal encoding was used in order to re-express “Yes” as 1 and “No” as 2 (Middleton, 2022). Lastly, outliers were detected by way of visual inspection using the Seaborn boxplot function for all quantitative variables described. This involved importing the seaborn package into the Jupyter notebook and using the appropriate command to plot the graph (Waskom, Ph.D, n.d.).

## C2. Reasoning

The functions and methodology used for the detections of duplicate values, missing values and outliers was selected by inspecting the pandas library documentation for any appropriate command that would achieve the desired outcome (NumFOCUS, Inc, 2023). Moreover, the PowerPoint slides provided in the course dashboard were used as a reference point for achieving the goal. The detection of duplicates and missing values was accomplished by incorporating the Python code specific to the *pandas* library, the boxplots were achieved using the specific *seaborn* package and the PCA analysis was achieved via the *SciKit* package.

## C3. Programming Language

The programming language used to clean the data was Python and its various libraries. This decision was based on prior experience in working with Python and the Anaconda package manager. Moreover, the IDE used was Visual Studio Code.

## C4. Code

For full code, see code attached.

# Returning a total count of duplicated values

df.duplicated().value\_counts()

# Returning a list of variables with total counts for missing values

df.isnull().sum()

#Plotting boxplots using seaborn package

sns.displot(df, x='Children')

sns.displot(df, x='Age')

sns.displot(df, x='Income')

sns.displot(df, x='Initial\_days')

# Part III: Data Cleaning (Treatment)

## D1. Detection Results

The first issue that was tackled was treating duplicated values. From the output in the code, we saw that there were no duplicated values detected in the entirety of the dataset. Since the output of the *df.duplicated().value\_counts()* function was ‘False 10000’, there were no duplicated values found in the whole data frame. Secondly, the variables with missing values were children, age, income, soft\_drink, overweight, anxiety and initial\_days and the values missing were 2588, 2414, 2464, 2467, 982, 984 and 1056 respectively. Outliers were also found in the dataset. For the quantitative variables, 4 outliers were found in the *Children* column, none were found in the *Age* and *Initial\_days* column and the variable with the most outliers was *Income*. Income’s 75th percentile quartile is 46466.7975 and the values above this (the outliers) were approximately 2500 records.

## D2. Treatment

Since there were no duplicated values, no further actions were taken. For the missing values, imputation was performed – for *Income, Children and Initial\_days,* the median was used as the reference value and for *Age* the mean was mean. Since *Income, Children and Initial\_days* were skewed to the right and  *Age* was uniformly distributed, the median value was selected (Straw, 2022).

Chart, histogram

Description automatically generated Chart, bar chart, histogram

Description automatically generated

Figure 4: Children Histogram (Original) Figure : Age Histogram (Original)

Chart, histogram

Description automatically generated Chart, histogram

Description automatically generated

Figure 6: Income Histogram (Original) Figure 7: Initial Days Histogram (Original)

After treatment, the histograms look like following:

Chart, histogram

Description automatically generated Chart, histogram

Description automatically generated

Figure 8: Children Histogram (Treated) Figure 9: Age Histogram (Treated)

Chart, histogram

Description automatically generated Chart, histogram

Description automatically generated

Figure 10: Initial\_days Histogram (Treated) Figure 11: Income histogram (Treated)

Outliers were first detected using seaborn boxplots as shown below.

Chart, box and whisker chart

Description automatically generated Chart

Description automatically generated

Figure 12: Children Boxplot (Original) Figure 13: Age Boxplot (Original)

*Chart, box and whisker chart

Description automatically generated* Chart

Description automatically generated

Figure 14: Income Boxplot (Original) Figure 15: Initial\_days Boxplot (Original)

The columns Age and Initial\_days had no outliers while Children had 4 and Income had 2500. Since we do not know if the outliers are factual errors, we will first extract the outliers, save them as their own dataframe and then remove them from the original data frame.

# Extracting records with z-scores -3 < z and z > 3 and saving as new variable

income\_outliers = df.query('Income\_z\_Scores < -3 | Income\_z\_Scores > 3')

children\_outliers = df.query('Children\_z\_Scores < -3 | Children\_z\_Scores > 3')

The above code shows how the outliers were extracted – first, the outliers were extracted into their own data frame, shown as *income\_outliers*  and *children\_outliers*. Secondly, the outliers were completely removed from the original data frame (*df*) and the new data frame was named *df\_new*, as shown in the code below.

# Creating a dataframe with Children and Income outliers removed and saving as df\_new

df\_new = df[(df['Income\_z\_Scores'] > -3) & (df['Income\_z\_Scores'] < 3) & (df['Children\_z\_Scores'] > -3) & (df['Children\_z\_Scores'] < 3)]

The z-scores were computed for the *Children* and *Income* variables; the first ten records of the z-scores are shown below:

Table

Description automatically generated

Figure 16: Children and Income z-scores

Lastly, the new data frame *df\_new* was checked for any outliers (effectively showing that the outliers were extracted successfully to their own dataframe). The code below checked for any outliers with a z-score of lower than -3 and higher than +3. Their sums were returned showing 0 (zero) for the sum of existing outliers.

Graphical user interface, text, application

Description automatically generated

Figure 17: Checking for Outliers after extraction

## D3. Summarized Work

The data after treatment contains no duplicated values. Moreover, the missing values were treated as well as the outliers. The code below shows that no missing values exist in the data after treatment:

Table

Description automatically generated Table

Description automatically generated

Figure 18: Output Showing No Missing Values

Moreover, outliers have been extracted to their own data frames so as not to completely remove these values from future analysis.

# Table of Figures

[Figure 1: Shape of Original Dataframe 5](#_Toc133352348)

[Figure 2: Duplicated Method 5](#_Toc133352349)

[Figure 3: Code Snippet for Detecting Missing Values 6](#_Toc133352350)

[Figure 4: Children Histogram (Original) ; Figure 5: Age Histogram (Original) 8](#_Toc133352351)

[Figure 6: Income Histogram (Original) ; Figure 7: Initial Days Histogram (Original) 8](#_Toc133352352)

[Figure 8: Children Histogram (Treated) ; Figure 9: Age Histogram (Treated) 9](#_Toc133352353)

[Figure 10: Initial\_days Histogram (Treated) ; Figure 11: Income histogram (Treated) 9](#_Toc133352354)

[Figure 12: Children Boxplot (Original) ; Figure 13: Age Boxplot (Original) 9](#_Toc133352355)

[Figure 14: Income Boxplot (Original) ; Figure 15: Initial\_days Boxplot (Original) 10](#_Toc133352356)

[Figure 16: Children and Income z-scores 11](#_Toc133352357)

[Figure 17: Checking for Outliers after extraction 12](#_Toc133352358)

[Figure 18: Output Showing No Missing Values 12](#_Toc133352359)

# Works Cited

Middleton, D. (2022). Webinar 3: Getting Started with Reexpression of Categorical Variables. Retrieved March 2023, from https://westerngovernorsuniversity.sharepoint.com/:b:/r/sites/D206DataCleaning/Shared%20Documents/Webinar%203%20Getting%20Started%20with%20Re-expression%20of%20Categorical%20Variables.pdf?csf=1&web=1&e=r3QoNq

NumFOCUS, Inc. (2023). *User Guide to pandas*, 1.53. Retrieved March 2023, from pandas: https://pandas.pydata.org/docs/user\_guide/index.html#user-guide

Straw, D. (2022). Data Cleaning: Webinar 2 - Getting Started with Missing Data and Outliers. Retrieved 2023, from https://westerngovernorsuniversity.sharepoint.com/sites/D206DataCleaning/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FD206DataCleaning%2FShared%20Documents%2FWebinar%202%20Getting%20Started%20with%20Missing%20Data%2Epdf&parent=%2Fsites%2FD206DataC

Waskom, Ph.D, M. (n.d.). *seaborn.boxplot*. Retrieved March 2023, from seaborn: statistical data visualization: https://seaborn.pydata.org/generated/seaborn.boxplot.html?highlight=boxplots