project_so_far

March 19, 2024

---

1 Performance Assessment: D208 Predictive Modeling Task 1 - Multiple Linear Regression.

1.1 Michael Hindes

Department of Information Technology, Western Governors University D208: Predictive Modeling Professor David Gagner February 11, 2024

This project aims to understanding the exact relationship between a response and predictor variables and to create a multiple regression model derived from medical raw data, targeting a business question reflective of a real-world organizational challenge. Python is employed to conduct a multiple regression analysis to explore the research question thoroughly. The analysis is supported by visual aids. The process also involves meticulous data cleaning to ensure accuracy and reliability. Additionally, the project shares the code used for the regression analysis, predictions and cleaning and wrangling. It concludes by detailing the regression equation, evaluating the statistical and practical significances, discussing limitations, and suggesting possible actions.

2 Part I: Research Question

2.1 Describe the purpose of this data analysis by doing the following::

2.1.1 A1. Research Question:

"A1. Research Question:"What factors contribute to the length of a patient's hospital stay?"

This question aims to identify key variables within the dataset that influence `Initial_days`, including possible financial aspects, services rendered, and patient risk factors. Understanding the primary drivers behind the length of a patient's hospital stay is crucial for multiple reasons. First, it allows healthcare providers to identify potential areas for improving operational efficiency, such as optimizing bed utilization and reducing wait times for incoming patients. Second, by recognizing which factors significantly impact hospital stay durations, healthcare professionals can tailor patient care plans more effectively, potentially leading to improved patient outcomes and satisfaction. Third, insights from this analysis can inform hospital administration decisions regarding resource allocation, staffing, and policy development, ensuring that the healthcare facility can better manage its resources while maintaining or enhancing the quality of care provided to patients. Lastly, given previous analyses on the dataset, there may be certain correlation between the number of days patient was hospitalized and his higher risk for readmission, which is costly to hospitals. Thus,

the goal of this research is not only to uncover the underlying factors affecting hospital stays but also to leverage this understanding to facilitate better hospital management practices and improve overall patient care.

2.1.2 A2. Define the goals of the data analysis.

This data analysis project is focused on developing a predictive model as a practical tool to help healthcare organizations in planning, patient care, and operational improvements. By examining a wide range of factors that potentially affect Initial_days, the project aims to understand any relationships initial_days may have with other variables. With that understanding, it seeks to build a model that supports data-driven decision-making in healthcare. The goals of the data analysis are as follows:

- Ensure Data Quality and Integrity: Prioritize data cleaning and preprocessing to ensure the dataset's accuracy, completeness, and consistency. This involves identifying and addressing missing values, outliers, and errors in the data. A clean dataset is fundamental for reliable analysis and modeling, enabling more accurate predictions and insights.
- Identify Key Predictors: Determine which variables significantly influence the length of hospital stays, considering a broad spectrum of factors, including but not limited to demographic information, medical history, financial aspects, and services received during the stay.
- Quantify Relationships: Establish how selected variables are related to Initial_days, quantifying the strength and nature of these relationships to provide actionable insights.
- Develop a Predictive Model: Create a multiple linear regression model that can accurately predict the length of a patient's hospital stay based on the identified variables, aiding in the anticipation of hospital capacity needs.
- Inform Policy Making: Provide evidence-based recommendations to healthcare administrators and policymakers for developing policies that address the key factors contributing to the length of hospital stays, with the goal of improving overall healthcare efficiency and patient satisfaction.

---

3 Part II: Method Justification

3.1 B. Describe multiple linear regression methods by doing the following:

3.1.1 B1. Summarize four assumptions of a multiple linear regression model:

In multiple linear regression analysis, four key assumptions are critical: linearity between variables, independence of observations, constant error variance (homoscedasticity), and normal distribution of error terms. Understanding and checking these assumptions is essential for the model's reliability and accuracy, providing a solid basis for predictive analytics.

- **Linearity** asserts that there is a straight-line relationship between each predictor (independent variable) and the response (dependent variable). This means that changes in a predictor variable are associated with proportional changes in the response variable.

- **Independence of Observations** indicates that the data points in the dataset do not influence each other. Each observation's response is determined by its predictor values, free from the effects of other observations in the dataset.
- **Homoscedasticity** refers to the requirement that the error terms (differences between observed and predicted values) maintain a consistent variance across all levels of the independent variables. This constant variance ensures that the model's accuracy does not depend on the value of the predictors.
- **Normality of Errors** involves the assumption that for any fixed value of an independent variable, the error terms are normally distributed. This normal distribution is central to conducting various statistical tests on the model's coefficients to determine their significance.

(Statology, n.d.) (Pennsylvania State University, n.d.)

3.1.2 B2. Describe two benefits of using Python for data analysis:

- **Rich Libraries:** While R was specifically designed with statistics and data analysis in mind, Python distinguishes itself with its comprehensive suite of libraries that cater to virtually every phase of the data analysis process. Libraries such as Pandas for data manipulation allow for efficient handling and transformation of data, NumPy for numerical computations supports complex mathematical operations with ease, and Matplotlib along with Seaborn for visualization enable the creation of insightful and high-quality graphs and charts. Moreover, Scikit-learn offers a robust platform for applying machine learning algorithms, streamlining the development of predictive models. These libraries not only facilitate a wide range of data analysis tasks but also ensure that analysts have the tools needed to tackle complex data challenges effectively.
- **Versatility** Python's syntax is known for its intuitive and readable nature, making it an accessible choice for professionals across various domains, from data science to web development. This versatility extends Python's utility beyond data analysis to other applications such as web development, automation, and deep learning, through frameworks and libraries like Flask, Selenium, and TensorFlow respectively. For instance, an analyst can easily switch from analyzing data to deploying a machine-learning model as a web application within the same programming environment. This seamless integration across different tasks enables a smooth workflow and promotes a holistic approach to problem-solving in today's interconnected digital landscape.

3.1.3 B3. Explain why multiple linear regression is an appropriate technique for analyzing the research question summarized in part I:

Multiple linear regression is particularly suited for addressing the research question at hand, as it facilitates the exploration of how several independent variables collectively influence a single continuous dependent variable, in this case, `Initial_days`. This analytical technique is adept at not only identifying but also quantifying the strength and nature of the relationships between Initial_days and various predictors, such as financial aspects, services rendered, and patient risk factors. By accounting for multiple factors simultaneously, multiple linear regression can provide nuanced insights into their combined effects on the length of a hospital stay. This comprehensive understanding is crucial for building robust predictive models that can inform decision-making processes.

4 NEEDS EDIT

Part III: Data Preparation

4.1 C. Summarize the data preparation process for multiple linear regression analysis by doing the following:

4.1.1 *C1. Describe your data cleaning goals and the steps used to clean the data to achieve the goals that align with your research question including your annotated code.**

- **Importing the Data**: Use`pd.read_csv()` to import data into a Pandas DataFrame.
- **Initial Data Examination**: Using `df.head()` provides a quick snapshot of the dataset, including a view of the first few rows. This helps in getting a preliminary understanding of the data's structure and content.
- **Checking Data Types**: The `df.info()` method is used for assessing the dataset's overall structure, including the data types of each column and the presence of non-null values.
- **Identifying Duplicate Rows**: Utilizing `df.duplicated()` to find duplicate rows is an essential cleaning step. Duplicates can skew your analysis and lead to inaccurate models. Once identified, you can decide whether to remove these rows with `df.drop_duplicates()` depending on their relevance to your research question.
- **Detecting Missing Values**: The `df.isnull().sum()` command is instrumental in identifying missing values across the dataset. Understanding where and how much data is missing is critical for deciding on imputation methods or if certain rows/columns should be excluded from the analysis.
- **Outlier Management Strategy:**: In this dataset, outliers are important to detect and be aware of. Particularly when creating predictive regression models. For example outliers can have a detrimental effect on our regression assumption and the model itself(MIDDLETON VIDEO) If there are outliers present, make sure that they are real is important. However, in the context of medical data, outliers can have an extra level of nuance. This is because in medical data, outliers are often the very things that we are interested in. For example, a patient with a very high cholesterol level or a very low blood pressure. These values are not errors, but rather important indicators of health conditions.
- Therefore, in this analysis, we will look for outliers and pay close attention to the details of the variables themselves. Outliers will be noted, but not necessarily treated unless they are obvious data entry errors or if they hinder our model.
- **Reviewing Unique Values**: Although `df.unique()` is used to explore unique values in a Series, for dataframes, you might consider `df.nunique()` to see the number of unique values in each column or use `df['column_name'].unique()` to check unique values in specific columns. This step is valuable for understanding the diversity of information within your dataset, particularly for categorical data.

-Wrangling

- **Drop Unnecessary Columns**: Any columns that are not relevant to the research question or the predictive model will be dropped from the dataset.

- **Categorical variable conversion**: Categorical variables will be transformed into numerical formats. Demographic data, which represents static information about patients and cannot be altered by the hospital, will be excluded from the analysis. We will identify and address any missing data, ensuring its proper mitigation. Additionally, any duplicate records identified in the dataset will be eliminated."

follow the slides here: https://westerngovernorsuniversity.sharepoint.com/:p:/r/sites/DataScienceTeam/_layouts/
8089-4758-9ABE-29976D079B56%7D&file=Dr.%20Sewell%20D208_Predictive_Modeling_Webinar_Episode%20

[ ]:
```
# Import packages and libraries
%pip install scikit-learn
%pip install Jinja2
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import seaborn as sns
from pandas import DataFrame
from sklearn import preprocessing
```

```
Requirement already satisfied: scikit-learn in
c:\users\hinde\appdata\local\programs\python\python312\lib\site-packages
(1.4.1.post1)
Requirement already satisfied: numpy<2.0,>=1.19.5 in
c:\users\hinde\appdata\local\programs\python\python312\lib\site-packages (from
scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in
c:\users\hinde\appdata\local\programs\python\python312\lib\site-packages (from
scikit-learn) (1.12.0)
Requirement already satisfied: joblib>=1.2.0 in
c:\users\hinde\appdata\local\programs\python\python312\lib\site-packages (from
scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\hinde\appdata\local\programs\python\python312\lib\site-packages (from
scikit-learn) (3.3.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 23.3.2 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Requirement already satisfied: Jinja2 in
c:\users\hinde\appdata\local\programs\python\python312\lib\site-packages
(3.1.2)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: MarkupSafe>=2.0 in
c:\users\hinde\appdata\local\programs\python\python312\lib\site-packages (from
Jinja2) (2.1.3)
```

```
[notice] A new release of pip is available: 23.3.2 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

- Display variable description and data types with examples.

[ ]:
```
# original data variable description and data types with examples.

from IPython.display import Image
Image(filename='variable_description.png')
```

[ ]:

Variable name	Data type	Variable Type	Description	Example
CaseOrder	int64	Numerical - Discrete	A variable to maintain the original sequence of the raw data file.	1
Customer_id	object	Categorical - Nominal	Distinct ID for each patient.	C412403
Interaction	object	Categorical - Nominal	Unique identifiers associated with patient interactions, operations, and hospitalizations.	8cd49b13-f45a-4b47-a2bd-173ffa932c2f
UID	object	Categorical - Nominal	Distinct identifiers linked to patient transactions, operations, and hospitalizations.	3a83ddb66e2ae73798bdf1d705dc0932
City	object	Categorical - Nominal	The city where the patient resides.	Mobile
State	object	Categorical - Nominal	The state where the patient resides.	AL
County	object	Categorical - Nominal	The county where the patient resides.	Morgan
Zip	int64	Categorical - Nominal	The zip code of the patient's residence.	35621
Lat	float64	Numerical - Continuous	Lattitudinal coordinates of the patient's home.	34.3496
Lng	float64	Numerical - Continuous	Longitudinal coordinates of the patient's home.	-86.72508
Population	int64	Numerical - Discrete	Number of people within a one-mile radius of the patient, as per census data.	2951
Area	object	Categorical - Nominal	Classification of area (suburban, urban, rural) according to unofficial census data.	Suburban
Timezone	object	Categorical - Nominal	Time zone of the patient's residence based on their registration information.	America/Chicago
Job	object	Categorical - Nominal	Occupation of the patient (or the primary insurance holder).	Psychologist, sport and exercise
Children	float64	Numerical - Discrete	Count of children in the patient's home.	1
Age	float64	Numerical - Discrete	Patient's age.	53
Education	object	Categorical - Ordinal	Patient's highest level of education achieved.	Some College, Less than 1 Year
Employment	object	Categorical - Nominal	Patient's current employment status.	Full Time
Income	float64	Numerical - Continuous	Yearly income of the patient (or the primary insurance holder).	86575.93
Marital	object	Categorical - Nominal	Patient's marital status (or the primary insurance holder).	Divorced
Gender	object	Categorical - Nominal	Patient's self-identified gender as male, female, or nonbinary.	Male
ReAdmis	object	Categorical - Binary	Indication of whether the patient was readmitted within a month of discharge (Yes, No).	No
VitD_levels	float64	Numerical - Continuous	Measurement of the patient's vitamin D levels in ng/mL.	17.80233049
Doc_visits	int64	Numerical - Discrete	Count of primary physician's visits to the patient during the first hospital stay.	6
Full_meals_eaten	int64	Numerical - Discrete	Count of complete meals consumed by the patient during hospitalization (partial meals are counted as 0).	0
VitD_supp	int64	Numerical - Discrete	Frequency of supplemental vitamin D administration to the patient.	0
Soft_drink	object	Categorical - Binary	Indication of whether the patient regularly consumes three or more sodas per day (Yes, No).	Yes
Initial_admin	object	Categorical - Nominal	The method of initial hospital admission for the patient (emergency admission, elective admission, observation).	Emergency Admission
HighBlood	object	Categorical - Binary	Indication of whether the patient has hypertension (Yes, No).	Yes
Stroke	object	Categorical - Binary	Indication of whether patient has experienced a stroke in past (Yes, No).	No
Complication_risk	object	Categorical - Ordinal	Patient's risk level for complications as determined by a primary patient assessment (high, medium, low).	Medium
Overweight	float64	Categorical - Binary	Specifies if patient is deemed overweight based on age, gender, and height (Yes, No).	0
Arthritis	object	Categorical - Binary	Specifies if patient has arthritis (Yes, No).	Yes
Diabetes	object	Categorical - Binary	Specifies if patient has diabetes (Yes, No).	Yes
Hyperlipidemia	object	Categorical - Binary	Specifies if patient has hyperlipidemia (Yes, No).	No
BackPain	object	Categorical - Binary	Specifies if patient suffers from chronic back pain (Yes, No).	Yes
Anxiety	float64	Categorical - Binary	Specifies if patient has an anxiety disorder (Yes, No).	1
Allergic_rhinitis	object	Categorical - Binary	Specifies if patient has allergic rhinitis (Yes, No).	Yes
Reflux_esophagitis	object	Categorical - Binary	Specifies if patient has reflux esophagitis (Yes, No).	No
Asthma	object	Categorical - Binary	Specifies if patient has asthma (Yes, No).	Yes
Services	object	Categorical - Nominal	Main service provided to the patient during hospitalization (blood work, intravenous, CT scan, MRI).	Blood Work
Initial_days	float64	Numerical - Continuous	Duration of the patient's initial hospital stay in days.	10.58576971
TotalCharge	float64	Numerical - Continuous	Daily charge to the patient. Figure represents the usual charges billed to patients, excluding specialized treatments.	3191.048774
Additional_charges	float64	Numerical - Continuous	Average charge to the patient for additional procedures, treatments, medications, anesthesiology, etc.	17939.40342
Item1	int64	Categorical - Ordinal	Prompt admission.	3
Item2	int64	Categorical - Ordinal	Timely care.	3
Item3	int64	Categorical - Ordinal	Regular visits.	2
Item4	int64	Categorical - Ordinal	Dependability.	2
Item5	int64	Categorical - Ordinal	Choices.	4
Item6	int64	Categorical - Ordinal	Treatment hours.	3
Item7	int64	Categorical - Ordinal	Polite staff.	3
Item8	int64	Categorical - Ordinal	Doctor's demonstration of active listening.	4

[ ]:
```
# import the data and read it into a dataframe, setting the first column␣
↪CaseOrder as the index

df = pd.read_csv('D208_templates/medical_clean.csv', index_col=0)
```

```
# Display the first five rows of the data
df.head()
```

[ ]:
```
           Customer_id                             Interaction  \
CaseOrder
1              C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
2              Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
3              F995323  a2057123-abf5-4a2c-abad-8ffe33512562
4              A879973  1dec528d-eb34-4079-adce-0d7a40e82205
5              C544523  5885f56b-d6da-43a3-8760-83583af94266

                                     UID          City State        County  \
CaseOrder
1           3a83ddb66e2ae73798bdf1d705dc0932           Eva    AL        Morgan
2           176354c5eef714957d486009feabf195      Marianna    FL       Jackson
3           e19a0fa00aeda885b8a436757e889bc9   Sioux Falls    SD     Minnehaha
4           cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca
5           d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William

               Zip       Lat       Lng  Population  …  TotalCharge  \
CaseOrder                                          …
1            35621  34.34960 -86.72508        2951  …  3726.702860
2            32446  30.84513 -85.22907       11303  …  4193.190458
3            57110  43.54321 -96.63772       17125  …  2434.234222
4            56072  43.89744 -93.51479        2162  …  2127.830423
5            23181  37.59894 -76.88958        5287  …  2113.073274

           Additional_charges Item1  Item2  Item3  Item4 Item5 Item6 Item7  \
CaseOrder
1                17939.403420     3      3      2      2     4     3     3
2                17612.998120     3      4      3      4     4     4     3
3                17505.192460     2      4      4      4     3     4     3
4                12993.437350     3      5      5      3     4     5     5
5                 3716.525786     2      1      3      3     5     3     4

            Item8
CaseOrder
1               4
2               3
3               3
4               5
5               3

[5 rows x 49 columns]
```

[ ]:
```
# View the last 5 rows of the dataframe
df.tail()
```

[ ]:
```
           Customer_id                                Interaction  \
CaseOrder
9996           B863060  a25b594d-0328-486f-a9b9-0567eb0f9723
9997           P712040  70711574-f7b1-4a17-b15f-48c54564b70f
9998           R778890  1d79569d-8e0f-4180-a207-d67ee4527d26
9999           E344109  f5a68e69-2a60-409b-a92f-ac0847b27db0
10000          I569847  bc482c02-f8c9-4423-99de-3db5e62a18d5

                                        UID         City State       County  \
CaseOrder
9996       39184dc28cc038871912ccc4500049e5      Norlina    NC       Warren
9997       3cd124ccd43147404292e883bf9ec55c       Milmay    NJ     Atlantic
9998       41b770aeee97a5b9e7f69c906a8119d7    Southside    TN   Montgomery
9999       2bb491ef5b1beb1fed758cc6885c167a        Quinn    SD   Pennington
10000      95663a202338000abdf7e09311c2a8a1   Coraopolis    PA    Allegheny

              Zip       Lat        Lng  Population  … TotalCharge  \
CaseOrder                                          …
9996        27563  36.42886  -78.23716        4762  …    6850.942
9997         8340  39.43609  -74.87302        1251  …    7741.690
9998        37171  36.36655  -87.29988         532  …    8276.481
9999        57775  44.10354 -102.01590         271  …    7644.483
10000       15108  40.49998  -80.19959       41524  …    7887.553

           Additional_charges Item1  Item2  Item3  Item4 Item5 Item6 Item7  \
CaseOrder
9996                 8927.642     3      2      2      3     4     3     4
9997                28507.150     3      3      4      2     5     3     4
9998                15281.210     3      3      3      4     4     2     3
9999                 7781.678     5      5      3      4     4     3     4
10000               11643.190     4      3      3      2     3     6     4

           Item8
CaseOrder
9996           2
9997           4
9998           2
9999           3
10000          3

[5 rows x 49 columns]
```

[ ]:
```
# Check the DataFrame information
df.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10000 entries, 1 to 10000
Data columns (total 49 columns):
```

```
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Customer_id         10000 non-null  object
 1   Interaction         10000 non-null  object
 2   UID                 10000 non-null  object
 3   City                10000 non-null  object
 4   State               10000 non-null  object
 5   County              10000 non-null  object
 6   Zip                 10000 non-null  int64
 7   Lat                 10000 non-null  float64
 8   Lng                 10000 non-null  float64
 9   Population          10000 non-null  int64
 10  Area                10000 non-null  object
 11  TimeZone            10000 non-null  object
 12  Job                 10000 non-null  object
 13  Children            10000 non-null  int64
 14  Age                 10000 non-null  int64
 15  Income              10000 non-null  float64
 16  Marital             10000 non-null  object
 17  Gender              10000 non-null  object
 18  ReAdmis             10000 non-null  object
 19  VitD_levels         10000 non-null  float64
 20  Doc_visits          10000 non-null  int64
 21  Full_meals_eaten    10000 non-null  int64
 22  vitD_supp           10000 non-null  int64
 23  Soft_drink          10000 non-null  object
 24  Initial_admin       10000 non-null  object
 25  HighBlood           10000 non-null  object
 26  Stroke              10000 non-null  object
 27  Complication_risk   10000 non-null  object
 28  Overweight          10000 non-null  object
 29  Arthritis           10000 non-null  object
 30  Diabetes            10000 non-null  object
 31  Hyperlipidemia      10000 non-null  object
 32  BackPain            10000 non-null  object
 33  Anxiety             10000 non-null  object
 34  Allergic_rhinitis   10000 non-null  object
 35  Reflux_esophagitis  10000 non-null  object
 36  Asthma              10000 non-null  object
 37  Services            10000 non-null  object
 38  Initial_days        10000 non-null  float64
 39  TotalCharge         10000 non-null  float64
 40  Additional_charges  10000 non-null  float64
 41  Item1               10000 non-null  int64
 42  Item2               10000 non-null  int64
 43  Item3               10000 non-null  int64
 44  Item4               10000 non-null  int64
 45  Item5               10000 non-null  int64
```

```
 46  Item6                10000 non-null  int64
 47  Item7                10000 non-null  int64
 48  Item8                10000 non-null  int64
dtypes: float64(7), int64(15), object(27)
memory usage: 3.8+ MB
```

[ ]:
```
# Check for duplicate rows.
print(df.duplicated().value_counts())

# Display the count of duplicate rows
print('Total Duplicated Rows: ', df.duplicated().sum())
```

```
False    10000
Name: count, dtype: int64
Total Duplicated Rows:  0
```

[ ]:
```
# Check for null values
df.isnull().sum()
```

[ ]:
```
Customer_id          0
Interaction          0
UID                  0
City                 0
State                0
County               0
Zip                  0
Lat                  0
Lng                  0
Population           0
Area                 0
TimeZone             0
Job                  0
Children             0
Age                  0
Income               0
Marital              0
Gender               0
ReAdmis              0
VitD_levels          0
Doc_visits           0
Full_meals_eaten     0
vitD_supp            0
Soft_drink           0
Initial_admin        0
HighBlood            0
Stroke               0
Complication_risk    0
Overweight           0
```

```
Arthritis             0
Diabetes              0
Hyperlipidemia        0
BackPain              0
Anxiety               0
Allergic_rhinitis     0
Reflux_esophagitis    0
Asthma                0
Services              0
Initial_days          0
TotalCharge           0
Additional_charges    0
Item1                 0
Item2                 0
Item3                 0
Item4                 0
Item5                 0
Item6                 0
Item7                 0
Item8                 0
dtype: int64
```

5 C2. EDA

[ ]:
```
# rename columns Item 1 to Item 8 to the appropriate column names. The 'S_'␣
 ↪modifier is used to indicate the column is a survey item.
new_col_names={
    'Item1':'S_T_Admission',
    'Item2':'S_T_Treatment',
    'Item3':'S_T_Visits',
    'Item4':'S_Reliability', 'Item5':'S_Options',
    'Item6':'S_Hours_Treatment',
    'Item7':'S_Staff',
    'Item8':'S_Active_Listening'}
df.rename(columns=new_col_names, inplace=True)
df.columns
```

[ ]:
```
Index(['Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip',
       'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Children',
       'Age', 'Income', 'Marital', 'Gender', 'ReAdmis', 'VitD_levels',
       'Doc_visits', 'Full_meals_eaten', 'vitD_supp', 'Soft_drink',
       'Initial_admin', 'HighBlood', 'Stroke', 'Complication_risk',
       'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
       'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma',
       'Services', 'Initial_days', 'TotalCharge', 'Additional_charges',
       'S_T_Admission', 'S_T_Treatment', 'S_T_Visits', 'S_Reliability',
```

```
       'S_Options', 'S_Hours_Treatment', 'S_Staff', 'S_Active_Listening'],
      dtype='object')
```

[ ]:
```
# combine the data types and unique values count into a DataFrame easy␣
 ↪reference and comparison
data_types = df.dtypes

unique_values = df.nunique()

comparison_df = pd.DataFrame({'Data Type': data_types, 'Unique Values':␣
 ↪unique_values})

comparison_df.sort_values(by='Unique Values', ascending=False)
```

[ ]:

	Data Type	Unique Values
Customer_id	object	10000
UID	object	10000
Interaction	object	10000
Initial_days	float64	9997
TotalCharge	float64	9997
Income	float64	9993
VitD_levels	float64	9976
Additional_charges	float64	9418
Lng	float64	8725
Zip	int64	8612
Lat	float64	8588
City	object	6072
Population	int64	5951
County	object	1607
Job	object	639
Age	int64	72
State	object	52
TimeZone	object	26
Children	int64	11
Doc_visits	int64	9
Full_meals_eaten	int64	8
S_T_Visits	int64	8
S_T_Admission	int64	8
S_Options	int64	7
S_Reliability	int64	7
S_T_Treatment	int64	7
S_Staff	int64	7
S_Hours_Treatment	int64	7
S_Active_Listening	int64	7
vitD_supp	int64	6
Marital	object	5
Services	object	4

Complication_risk	object	3
Area	object	3
Gender	object	3
Initial_admin	object	3
Asthma	object	2
Reflux_esophagitis	object	2
Overweight	object	2
Diabetes	object	2
Stroke	object	2
HighBlood	object	2
Soft_drink	object	2
Allergic_rhinitis	object	2
ReAdmis	object	2
Anxiety	object	2
BackPain	object	2
Hyperlipidemia	object	2
Arthritis	object	2

6 Cardinality and Data Type Summary of Variables

6.1 Ratio Variables (Continuous with an absolute zero)

- Income: 9993 unique values (float64)
- VitD_levels: 9976 unique values (float64)
- Initial_days: 9997 unique values (float64)
- TotalCharge: 9997 unique values (float64)
- Additional_charges: 9418 unique values (float64)
- Population: 5951 unique values (int64)
- Children: 11 unique values (int64)
- Age: 72 unique values (int64)
- Doc_visits: 9 unique values (int64)
- Full_meals_eaten: 8 unique values (int64)
- vitD_supp: 6 unique values (int64)

6.2 Interval Variables (Continuous without an absolute zero)

- Lat: 8588 unique values (float64)
- Lng: 8725 unique values (float64)

6.3 Ordinal Variables

- S_T_Admission: 8 unique values (int64)
- S_T_Treatment: 7 unique values (int64)
- S_T_Visits: 8 unique values (int64)
- S_Reliability: 7 unique values (int64)
- S_Options: 7 unique values (int64)
- S_Hours_Treatment: 7 unique values (int64)
- S_Staff: 7 unique values (int64)

- `S_Active_Listening`: 7 unique values (int64)

6.4 Nominal Variables (Categorical)

- `Customer_id`: 10000 unique values (object)
- `Interaction`: 10000 unique values (object)
- `UID`: 10000 unique values (object)
- `City`: 6072 unique values (object)
- `State`: 52 unique values (object)
- `County`: 1607 unique values (object)
- `Zip`: 8612 unique values (int64)
- `Area`: 3 unique values (object)
- `TimeZone`: 26 unique values (object)
- `Job`: 639 unique values (object)
- `Marital`: 5 unique values (object)
- `Gender`: 3 unique values (object)
- `ReAdmis`: 2 unique values (object)
- `Soft_drink`: 2 unique values (object)
- `Initial_admin`: 3 unique values (object)
- `HighBlood`: 2 unique values (object)
- `Stroke`: 2 unique values (object)
- `Complication_risk`: 3 unique values (object)
- `Overweight`: 2 unique values (object)
- `Arthritis`: 2 unique values (object)
- `Diabetes`: 2 unique values (object)
- `Hyperlipidemia`: 2 unique values (object)
- `BackPain`: 2 unique values (object)
- `Anxiety`: 2 unique values (object)
- `Allergic_rhinitis`: 2 unique values (object)
- `Reflux_esophagitis`: 2 unique values (object)
- `Asthma`: 2 unique values (object)
- `Services`: 4 unique values (object)

Given the nature of the data, there are several variables that will be excluded from the analysis. Here is a brief summary of the variables that will be excluded and the rationale for their exclusion:

6.4.1 Current Strategy Overview:

1. **Broad Inclusion**: Start with a wide array of variables to capture potential influences on `Initial_days`.
2. **Build Initial Model**: Use this extensive dataset to identify significant predictors.
3. **Analyze & Refine**: Eliminate non-contributing or highly correlated variables based on initial model insights.
4. **Develop Reduced Model**: Focus on key variables for a streamlined, effective model.

6.4.2 Variables Eliminated:

- **TotalCharge & Additional Charges**: Possible high correlation and generally a result of `Initial_days` not a cause of.
- **Latitude & Longitude**: Limited interpretive value.
- **Identifiers (Customer_id, Interaction, UID)**: High uniqueness; ethical concerns.
- **Geographic (City, State, County, Zip)**: Overly detailed, increasing model complexity.
- **TimeZone**: Relevance to hospital stay length is questionable.
- **Job**: Subjective and variable in interpretation.

[ ]:
```
# create reduced dataframe with only the columns  for the analysis
# columns to drop
colms_to_drop = ['TotalCharge', 'Additional_charges', 'Lat', 'Lng',␣
 ↪'Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip',␣
 ↪'TimeZone', 'Job']

# Creates list of columns except the 'Initial_days'
remaining_cols = [col for col in df.columns if col not in colms_to_drop +␣
 ↪['Initial_days']]

# Addd 'Initial_days' to the end of column list - for backwards elimination␣
 ↪intuition
final_cols = remaining_cols + ['Initial_days']

# crerate reduced_df with the proper column order
df_reduced = df[final_cols].copy()

# display the dataframe
df_reduced
```

[ ]:

CaseOrder	Population	Area	Children	Age	Income	Marital	Gender	…	ReAdmis	VitD_levels	Doc_visits	…	Services	S_T_Admission	…
1	2951	Suburban	1	53	86575.93	Divorced	Male		No	19.141466	6	…	Blood Work	3	
2	11303	Urban	3	51	46805.99	Married	Female		No	18.940352	4	…	Intravenous	3	
3	17125	Suburban	3	53	14370.14	Widowed	Female								
4	2162	Suburban	0	78	39741.49	Married	Male								
5	5287	Rural	1	22	1209.56	Widowed	Female								
…	…	…	…	…	…	…	…								
9996	4762	Urban	2	25	45967.61	Widowed	Male								
9997	1251	Urban	4	87	14983.02	Widowed	Male								
9998	532	Rural	3	45	65917.81	Separated	Female								
9999	271	Rural	3	43	29702.32	Divorced	Male								
10000	41524	Urban	8	70	62682.63	Separated	Female								

```
3              No    18.057507           4  …   Blood Work              2
4              No    16.576858           4  …   Blood Work              3
5              No    17.439069           5  …      CT Scan              2
…             …           …          …   …          …              …
9996           No    16.980860           4  …  Intravenous             3
9997          Yes    18.177020           5  …      CT Scan              3
9998          Yes    17.129070           4  …  Intravenous             3
9999          Yes    19.910430           5  …   Blood Work              5
10000         Yes    18.388620           5  …   Blood Work              4

           S_T_Treatment S_T_Visits S_Reliability S_Options S_Hours_Treatment  \
CaseOrder
1                      3          2             2         4                 3
2                      4          3             4         4                 4
3                      4          4             4         3                 4
4                      5          5             3         4                 5
5                      1          3             3         5                 3
…                    …          …             …         …               …
9996                   2          2             3         4                 3
9997                   3          4             2         5                 3
9998                   3          3             4         4                 2
9999                   5          3             4         4                 3
10000                  3          3             2         3                 6

           S_Staff S_Active_Listening Initial_days
CaseOrder
1                3                   4    10.585770
2                3                   3    15.129562
3                3                   3     4.772177
4                5                   5     1.714879
5                4                   3     1.254807
…              …                   …          …
9996             4                   2    51.561220
9997             4                   4    68.668240
9998             3                   2    70.154180
9999             4                   3    63.356900
10000            4                   3    70.850590

[10000 rows x 36 columns]
```

[ ]:
```
# Summary Stats For ratio and interval variables (numerical summary)
selected_columns = df_reduced[['Population', 'Age', 'Income', 'VitD_levels',␣
↪'Doc_visits', 'Full_meals_eaten', 'vitD_supp', 'Initial_days']].copy()
selected_columns.describe()
```

[ ]:
```
          Population           Age         Income   VitD_levels    Doc_visits  \
count   10000.000000  10000.000000   10000.000000  10000.000000  10000.000000
```

```
mean     9965.253800     53.511700   40490.495160     17.964262      5.012200
std     14824.758614     20.638538   28521.153293      2.017231      1.045734
min         0.000000     18.000000     154.080000      9.806483      1.000000
25%       694.750000     36.000000   19598.775000     16.626439      4.000000
50%      2769.000000     53.000000   33768.420000     17.951122      5.000000
75%     13945.000000     71.000000   54296.402500     19.347963      6.000000
max    122814.000000     89.000000  207249.100000     26.394449      9.000000

       Full_meals_eaten     vitD_supp  Initial_days
count      10000.000000  10000.000000  10000.000000
mean           1.001400      0.398900     34.455299
std            1.008117      0.628505     26.309341
min            0.000000      0.000000      1.001981
25%            0.000000      0.000000      7.896215
50%            1.000000      0.000000     35.836244
75%            2.000000      1.000000     61.161020
max            7.000000      5.000000     71.981490
```

6.4.3 Initial Takeaways:

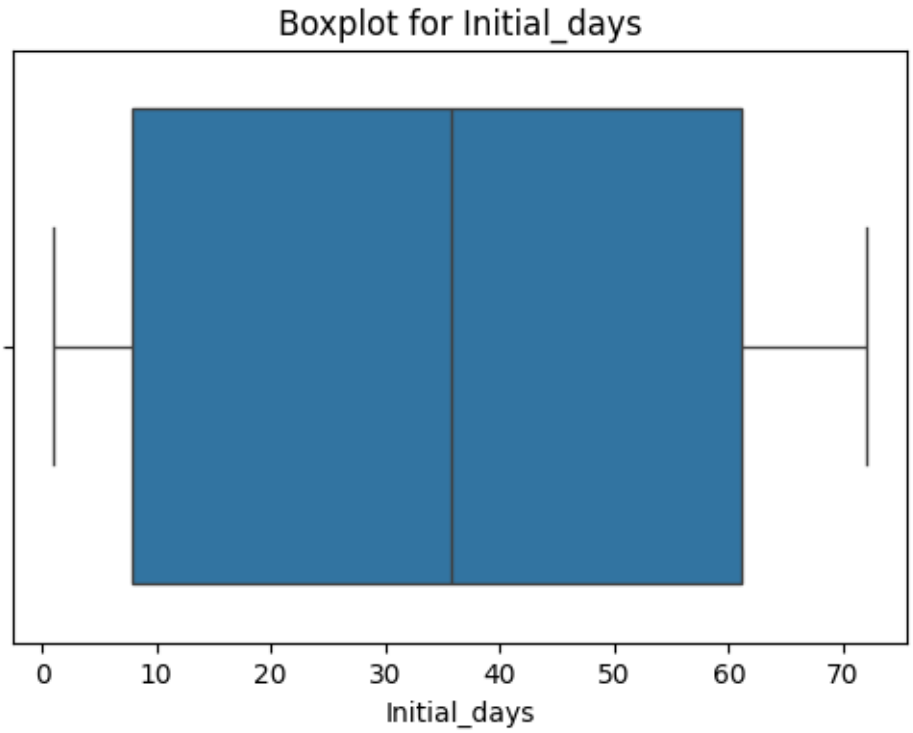
- **Population**: Averages 9965 with a broad range (0 to 122814), suggesting varied area sizes.
- **Age**: Averages 53 years, ranging from 18 to 89, with a diverse age profile.
- **Income**: Averages $40,490, with wide variation (154 to 207249), indicating economic diversity.
- **VitD_levels**: Averages 17.96, mostly within a narrow range (9.81 to 26.39), suggesting more consistent levels across patients.
- **Doc_visits**: Averages 5 visits, indicating a similar frequency of medical consultations.
- **Full_meals_eaten**: Averages slightly over 1 meal a day, with most patients having similar meal frequencies.
- **vitD_supp**: Averages less than 0.5 supplements, with low intake common among patients.
- **Categorical** nominal and ordinal variables are not included here and will include a separate summary of proportions along wit univariate and bivariate visualizations.
- **Initial_days**: Our dependent (target) variable will be fully summarize and visualized below

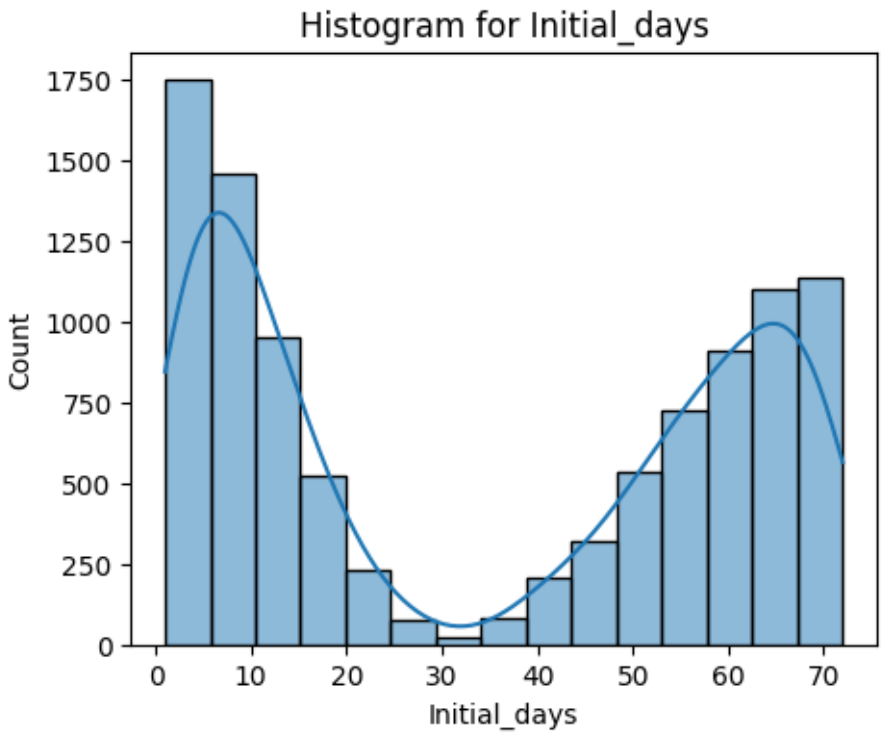
[ ]:
```
# To CSV before transform
df_reduced.to_csv('df_reduced1.csv', index=False)
```

[ ]:
```
# Boxplot for 'Initial_days'
plt.figure(figsize=(6, 4))
sns.boxplot(x=df_reduced['Initial_days'])
plt.title('Boxplot for Initial_days')
plt.show()
```

```
# Histogram for 'Initial_days'
plt.figure(figsize=(5, 4))
sns.histplot(data=df_reduced, x='Initial_days', kde=True)
plt.title('Histogram for Initial_days')
plt.show()

df_reduced['Initial_days'].describe()
```






```
[ ]: count    10000.000000
     mean        34.455299
     std         26.309341
     min          1.001981
     25%          7.896215
     50%         35.836244
     75%         61.161020
     max         71.981490
     Name: Initial_days, dtype: float64
```

- **Boxplot Observations**: The median appears to be above the mid-30s, suggesting that roughly half of the patients have shorter initial stays and the other half have longer. There are no visible outliers, indicating no extreme values or anomalies that fall outside the typical range. The interfertile range shows that the middle 50% of the data spans a rather large range, suggesting a concentration of data within this segment.
- **Histogram Observations**: The distribution is bimodal, with two peaks: one just under a few days and another around 70 days. This suggests there are two groups of patients with different typical hospital stay lengths. The histogram indicates that shorter initial stays are more common than longer stays, with a significant drop-off in frequency as the number of days increases towards the middle values. The spread between the two modes shows that there is variability in the data, not concentrated around a single central value. The bimodal distribution could imply two prevalent groups or clusters within the dataset for `Initial_days`. Understanding the reasons behind this bimodal distribution may require further investigation

into the underlying factors affecting hospital stay lengths. This distribution is important to kee in mind when interpreting the results of the regression analysis, as it may influence the model's predictive accuracy and the significance of the predictors.

Summary: Statistical measures for `Initial_days` across all patients in the dataset, including:

- **Count**: 10,000 observations. This represents the number of patients included in the analysis.
- **Mean**: Approximately 34 days. On average, patients spend a little over a month in the hospital.
- **Standard Deviation**: About 26 days. This indicates a wide variation in the length of hospital stays among patients; while some patients have short stays, others have significantly longer stays.
- **Minimum**: Just over 1 day. This shows that some patients are discharged almost immediately after admission.
- **25% (First Quartile)**: About 8 days or less. A quarter of the patients have hospital stays just over a week.
- **Median (50%)**: Approximately 36 days. This is very close to the mean. However, the slight difference between the mean and median indicates a slight skew in the data.
- **75% (Third Quartile)**: About 61 days or less. Most patients are discharged within two months.
- **Maximum**: Nearly 72 days. Indicates that some patients have extended hospital stays.

---

7 C3. Visualizations

7.0.1 G & H: References

- Western Governors University. (2023, December 21). D207 - Medical_clean Dataset. Retrieved from https://lrps.wgu.edu/provision/227079957
- Western Governors University IT Department. (2023). R or Python? How to decide which programming language to learn. Retrieved from https://www.wgu.edu/online-it-degrees/programming-languages/r-or-python.html#
- Datacamp. (2023, December 12). D207 - Exploratory Data Analysis. Retrieved from https://app.datacamp.com/learn/custom-tracks/custom-d207-exploratory-data-analysis
- Sewell, Dr. (2023). WGU D207 Exploratory Data Analysis [Webinars]. WGU Webex. Accessed December, 2023. https://wgu.webex.com/webappng/sites/wgu/meeting/info/c4aca2eac546482880f1557c938abf40?siteurl=wg