D208_complete

March 26, 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 Dr. Straw March 21, 2024

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 initial hospital stay?"

This question aims to identify key variables within the dataset that influence `Initial_days`; The number of days the patient stayed in the hospital during the initial visit to the hospital.

2.1.2 A2. Define the goals of the data analysis.

The project sets out to explore the relationship between a response and predictor variables by exploring raw medical data and developing a multiple linear regression model. The research question focuses on identifying any potential factors that affect the length of a patient's initial hospital stay by exploring factors such as demographic details, medical history, financial factors, and services received. Python and associated libraries are used for analysis, and that supported by visual aids for clarity. Data cleaning and wrangling is emphasized to ensure accuracy and reliability.The Python code for analysis, data cleaning, and preparation will be shared. The culmination of this project involves creating, evaluating and reducing a multiple linear regression model, discussing its significance both statistically and practically, highlighting limitations, and suggesting actionable steps for stakeholders and future analysts based on the findings. Length of hospital stay is a critical metric in healthcare, as it can impact resource allocation, patient satisfaction, and overall hospital efficiency. By identifying the factors that contribute to a patient's hospital stay, healthcare providers can optimize their services, improve patient outcomes, and enhance the overall quality of care.

---

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 the research on the assumption of multiple linear regression, five key assumptions were found in some places and four in others, in different combinations. They all appear critical to the validity of a model. As such I will list five assumptions below. (Statology 2023)

- **Linearity** asserts that there is a straight-line relationship between each predictor (independent variable) and the response (dependent variable). In other words, a straight line can best show the average change in a dependent variable for one unit of change in the independent variable, holding all other independent variables constant. This can be assessed through visualizations.
- **Little to no Multicollinearity** In an ideal scenario, the explanatory variables in the dataset should not significantly influence each other. Each observation's response should be primarily determined by its own predictor values and should be minimally affected by the values of other independent variables.
- **Independence of Observations** Assumes the observations in the dataset be independent of each other, meaning that the value of one observation should not be influenced by the value of another. Violations may occur in temporal or clustered data. If violated, it can lead to biased standard errors and incorrect inferences. Independence can be assessed using residual plots or statistical tests like the Durbin-Watson test.
- **Homoscedasticity** refers to the requirement that the error terms (differences between observed and predicted values) maintain a constance variance across all points. This constant variance ensures that the model's accuracy does not depend on the value of the predictors. Homoscedasticity is often checked with a residuals plots to look for patterns where there should be none, and can be caused by a variety of factors.
- **Normality of Errors** states that the residuals (errors) in the model are normally distributed around a mean of zero. This can be checked with a histogram or Q-Q plot of the residuals. If the residuals are not normally distributed, the model may not be accurate.

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 was chosen for its suite of libraries that facilitate every phase of the data analysis process. Libraries such as Pandas for data manipulation, NumPy for numerical computations, and Matplotlib along with Seaborn for visualizations. Statsmodels and Scikit-learn offers a platforms for applying regression and machine learning algorithms, streamlining the development of predictive models. These libraries help with a range of data analysis tasks. (Western Governors University)
- **Versatility** Python's syntax is known for its intuitiveness and readability, and wide ranging application, making it a favorite for many, from data science to web development. This versatility extends beyond data analysis to other applications such as web development, automation, and deep learning. 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 flexibility is a significant advantage for working across multiple domains. (Western Governors University)

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 (MLR) is an appropriate statistical technique for addressing the research question at hand. Unlike simple linear regression, MLR can handle multiple explanatory or predictor (independent) variables, which is necessary for this case. MLR uses these variables to predict the outcome of a response or target (dependent) variable, which in this case is `Initial_days`, representing the length of a patient's initial stay in the hospital. This analytical method is good at identifying and quantifying the strength and nature of the relationships between `Initial_days` and various predictors. By accounting for multiple factors simultaneously, MLR provides more nuanced insights into their combined effects on the length of a hospital stay. This is essential for creating a predictive model that can effectively inform decision-making processes and help understand the key factors influencing patient outcomes. MLR's ability to handle multiple variables makes it an appropriate tool for analyzing datasets like `medical_clean.csv` and uncovering meaningful patterns that can guide healthcare strategies and interventions.

4 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.

The cleaning process starts by reading the data into a pandas DataFrame and performing an initial examination to gain a preliminary understanding of its structure and content. This involves checking data types, identifying duplicate rows, and detecting missing values. Outliers are important to detect and be aware of, particularly when creating predictive regression models. In the context of medical data, outliers can often be the very things that are of interest, such as patients with very high cholesterol levels or very low blood pressure. These values are not necessarily errors but rather important indicators of health conditions. Therefore, outliers will be noted but not necessarily treated unless they are obvious data entry errors or if they hinder the model.

Unique values will be examined to understand the variety of information within the dataset, dropping unnecessary columns that are not relevant to the research question or predictive model, and converting categorical variables into numerical formats. Some demographic and identifier data, which represents static information about patients and cannot be altered by the hospital, will be excluded from the analysis. Missing data will be identified and addressed, ensuring its proper mitigation, and any duplicate records will be eliminated. Renaming of certain variables for a more descriptive understanding. Rounding data to a reasonable number of decimal places can improve readability and reduce computational complexity. Data visualizations such as scatter plots, histograms, and box plots will be used to understand the relationships between variables and identify patterns in the data distribution.

4.1.2 The following requirements from `Part C` of the performance assessment will be demonstrated in the multiple cells below.

- **C2. Describe the dependent variable and all independent variables using summary statistics that are required to answer the research question.**
- **C3. Generate univariate and bivariate visualizations of the distributions of the dependent and independent variables.**
- **C4. Describe your data transformation goals that align with your research question**

(The Python code used in the project was heavily informed and sometimes directly puulled from the documentation listed in the `Software` *section of the References section concluding this project)*

```
[ ]: # Import packages and libraries
     %pip install scikit-learn
     %pip install Jinja2
     %matplotlib inline
     %pip install statsmodels
     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
     import statsmodels.api as sm
     from statsmodels.stats.outliers_influence import variance_inflation_factor
     from statsmodels.tools.tools import add_constant
```

```
[ ]: # original data variable description and data types with examples.
     from IPython.display import Image
     Image(filename='variable_description_208.png')
```

```
[ ]: # import the data and read it into a dataframe, setting the first column␣
      ↪`CaseOrder` as the index for consistency.
     df_medical = pd.read_csv('D208_templates/medical_clean.csv', index_col=0)

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

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

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

```
[ ]: # Check for duplicate rows.
     print(df_medical.duplicated().value_counts())
     print('Total Duplicated Rows: ', df_medical.duplicated().sum())
```

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

```
[ ]: # 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_medical.rename(columns=new_col_names, inplace=True)
     df_medical.columns
```

```
[ ]: # combine the data types and unique values count into a DataFrame easy␣
      ↪reference and comparison
     data_types = df_medical.dtypes
     unique_values = df_medical.nunique()
     comparison_df = pd.DataFrame({'Data Type': data_types, 'Unique Values':␣
      ↪unique_values})
     comparison_df.sort_values(by='Unique Values', ascending=False)
```

5 Cardinality and Data Type Summary of Variables

5.1 Numerical Variables

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

5.2 Ordinal Variables (Categorical)

- `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)

5.3 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:

5.3.1 Current Strategy Overview:

1. **Broad Inclusion**: *Cast a wide net* (Middleton, 2024) Start with a wide array of variables to capture potential influences on `Initial_days`, informed by domain knowledge and based on the reccomendation of the instructors of this course.
2. **Build Initial Model**: Use this dataset to identify significant variables.
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.

5.3.2 Variables Eliminated:

Note: I am a former health care professional who has worked in several hospitals, and unfortunatly have had extensive hospital stays as a patient as well. While I am not an expert on this particular data, I do have some domain knowledge, and this domain knowledge helps inform some of my decision making here. - **TotalCharge & Additional Charges**: Possible high correlation and generally a result of `Initial_days` not a cause of. Patients and staff often unaware of these charges until after the fact. - **Latitude & Longitude**: Limited interpretive value and adds to model complexity. - **Identifiers (Customer_id, Interaction, UID)**: High uniqueness; ethical concerns. - **Geographic (City, State, County, Zip, Population)**: Overly detailed, increasing model complexity, not short/medium term actionable. - **TimeZone**: Relevance to hospital stay length is questionable, increases complexity. - **Full_meals_eaten**: Restrictive and targeted diets and meals are so common and depends on patient and services that without context ths variable is not useful. - **Job**: Subjective and variable in interpretation. Better suited for targeted occupational study. - **Services**: All very common in diagnostic phase and itself dependent on too many unknown factors, and not likely to be significant predictors. Could add confusion. - **Soft_drink**: Poorly defined as soft drink can mean anything from uncaffinated carbonated water to a caffinated sugary soda.

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

df_reduced = df_medical.drop(colms_to_drop, axis=1)

# display the dataframe in full
pd.set_option('display.max_columns', None)
df_reduced.head().transpose()
```

```
[ ]: # Summary Stats For numeric variables
selected_columns = df_reduced[['Age', 'Income', 'Children', 'VitD_levels',␣
 ↪'Doc_visits', 'vitD_supp', 'Initial_days']].copy()
selected_columns.describe()
```

5.3.3 Initial Takeaways:

- **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.
- **Children**: Averages 2 children with a similar median, with a range of 0 to 10.
- **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.
- **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

6 Rounding Justification.

- Rounding `'Initial_days'` from 8 decimal places to 2 reduces the number of unique values, which can simplify analyses and visualizations by reducing the granularity of the data. Precision beyond 2 decimal places for representing days does not add meaningful information for the analysis. In many practical scenarios, especially related to days, a precision of 2 decimal places is sufficient to capture relevant variations without unnecessarily complicating the dataset. In healthcare data, for instance, it's unlikely that fractions of a day to eight decimal places would impact decisions or care outcomes.
- Similarly, rounding `Income` to whole numbers, and `'VitD_levels'` to 2 decimal places seems appropriate in this context.

```
[ ]: # round 'Initial_days' and 'VitD_levels' to 2 decimal places
     df_reduced = df_reduced.round({'VitD_levels': 2})
     df_reduced = df_reduced.round({'Initial_days': 2})

     # round 'Income' to 0 decimal places by converting to integer
     df_reduced = df_reduced.astype({'Income': 'int64'})

     # fisplay the dataframe with the rounded values
     df_reduced[['Initial_days', 'VitD_levels', 'Income']].head()
```

```
[ ]: # Export to csv and to save results so far and to reduce memory consumption.
     df_reduced.to_csv('df_reduced.csv', index='CaseOrder')
```

```
[ ]: # Load the data
     df = pd.read_csv('df_reduced.csv', index_col=0)
```

---

7 C3. Visualizations

Below are Univariate and Bivariate Visualizations for explanatory variables showing their relationship with the dependent variable `Initial_days`. Seaborn and Matplotlib will be used to create visualizations and the choice of graph will depend on the nature of the variable being visualized. (Python Graph Gallery. n.d), (Eyre, 2024)

8 Univaraite Visualizations

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

# Histogram for 'Initial_days'
plt.figure(figsize=(5, 4))
sns.histplot(data=df, x='Initial_days', kde=True, bins=50)
plt.title('Histogram for Initial_days')
plt.show()

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

- **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. Understanding the reasons behind this bimodal distribution may require further investigation into the 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.

[ ]:
```python
# subplots for the histplots
fig, axes = plt.subplots(1, 3, figsize=(10, 3))

# distribution/count of children
sns.histplot(data=df, x='Children', ax=axes[0], kde=True)
axes[0].set_title('Distribution/Count of Children')

# distribution of ages
sns.histplot(data=df, x='Age', ax=axes[1], kde=True)
axes[1].set_title('Distribution of Ages')

# distribution of income
sns.histplot(data=df, x='Income', ax=axes[2], kde=True)
axes[2].set_title('Distribution of Income')

plt.tight_layout()
plt.show()

# summary statistics for the variables
df[['Children', 'Age', 'Income']].describe().transpose()
```

[ ]:
```python
# subplots for the boxplots
fig, axes = plt.subplots(1, 3, figsize=(10, 3))

# boxplot of children
sns.boxplot(data=df, x='Children', ax=axes[0])
axes[0].set_title('Boxplot of Children')

# boxplot of ages
sns.boxplot(data=df, x='Age', ax=axes[1])
axes[1].set_title('Boxplot of Ages')

# boxplot of income
sns.boxplot(data=df, x='Income', ax=axes[2])
axes[2].set_title('Boxplot of Income')

plt.tight_layout()
plt.show()
```

- The outliers here will be noted as they may impact the regression model, particularly with OLS regression. For now, we will note them and include as is in the initial model.

[ ]:
```
# subplots for the histplots
fig, axes = plt.subplots(1, 3, figsize=(10, 3))

# distribution/count of VitD_levels
sns.histplot(data=df, x='VitD_levels', ax=axes[0], kde=True)
axes[0].set_title('Distribution of VitD_levels')

# distribution/count of Doc_visits with bigger bins
sns.histplot(data=df, x='Doc_visits', ax=axes[1], kde=True)
axes[1].set_title('Distribution/Count of Doc_visits')

# distribution/count of vitD_supp with bigger bins
sns.histplot(data=df, x='vitD_supp', ax=axes[2], kde=True)
axes[2].set_title('Distribution/count of vitD_supp')

plt.tight_layout()
plt.show()
# descriptive statistics for the variables
df[['VitD_levels', 'Doc_visits', 'vitD_supp']].describe().transpose()
```

- The `Vitamin D levels` appear normally distributed around a middle value, suggesting that most patients have Vitamin D levels within a standard range, with fewer individuals having very high or very low levels. `Doc_visits` show a pattern with most patientss having 4-6 visits, and the frequency drops for higher numbers of visits. For Vitamin `D supplements`, most patients are not given supplements, which aligns with the distribution of Vitamin D levels.

[ ]:
```
# Create a 2 by 2 subplot grid
fig, axes = plt.subplots(2, 2, figsize=(10, 6))

# Area
sns.countplot(data=df, x='Area', ax=axes[0, 0])
axes[0, 0].set_title('Count of Area')

# Marital
sns.countplot(data=df, x='Marital', ax=axes[0, 1])
axes[0, 1].set_title('Count of Marital Status')

# Gender
sns.countplot(data=df, x='Gender', ax=axes[1, 0])
axes[1, 0].set_title('Count of Gender')

# ReAdmis
```

```
sns.countplot(data=df, x='ReAdmis', ax=axes[1, 1])
axes[1, 1].set_title('Numer of ReAdmissions')

plt.tight_layout()
plt.show()

# create a countplot for initial_admin
plt.figure(figsize=(5, 4))
sns.countplot(data=df, x='Initial_admin')
plt.title('Count of Initial_admin')
plt.show()
```

9 Proportion Summary

`Area` - Rural: 33.69% - Urban: 33.03% - Suburban: 33.28%

`Gender` - Female: 50.18% - Male: 47.68% - Nonbinary: 2.14%

`Marital` - Widowed: 20.45% - Married: 20.23% - Separated: 19.87% - Never Married: 19.84% - Divorced: 19.61%

`ReAdmis` - No: 63.31% - Yes: 36.69%

`Initial_admin` - Emergency: 51.60% - Elective: 25.04% - Observation: 24.36%

[ ]:
```
# Survey items
# 2 by 4 subplot grid
fig, axes = plt.subplots(2, 4, figsize=(12, 6))

# S_T_Admission
sns.countplot(data=df, x='S_T_Admission', ax=axes[0, 0])
axes[0, 0].set_title('Count of S_T_Admission')

# S_T_Treatment
sns.countplot(data=df, x='S_T_Treatment', ax=axes[0, 1])
axes[0, 1].set_title('Count of S_T_Treatment')

# S_T_Visits
sns.countplot(data=df, x='S_T_Visits', ax=axes[0, 2])
axes[0, 2].set_title('Count of S_T_Visits')

# S_Reliability
sns.countplot(data=df, x='S_Reliability', ax=axes[0, 3])
axes[0, 3].set_title('Count of S_Reliability')

# S_Options
sns.countplot(data=df, x='S_Options', ax=axes[1, 0])
axes[1, 0].set_title('Count of S_Options')
```

```
# S_Hours_Treatment
sns.countplot(data=df, x='S_Hours_Treatment', ax=axes[1, 1])
axes[1, 1].set_title('Count of S_Hours_Treatment')

# S_Staff
sns.countplot(data=df, x='S_Staff', ax=axes[1, 2])
axes[1, 2].set_title('Count of S_Staff')

# S_Active_Listening
sns.countplot(data=df, x='S_Active_Listening', ax=axes[1, 3])
axes[1, 3].set_title('Count of S_Active_Listening')

plt.tight_layout()
plt.show()

# value counts for the survey items
df[['S_T_Admission', 'S_T_Treatment', 'S_T_Visits', 'S_Reliability',␣
↪'S_Options', 'S_Hours_Treatment', 'S_Staff', 'S_Active_Listening']].apply(pd.
↪Series.value_counts)
```

- Survey responses across various rating scales appear to be fairly evenly distributed among the different survey items. This uniformity could indicate a degree of correlation among the responses to these items. To explore potential patterns, we will utilize pie charts to visualize the distribution of responses and a correlation matrix to quantitatively assess the relationships between the items.

[ ]:
```
textprops = {"fontsize":10}

# 2 by 4 subplot grid
fig, axes = plt.subplots(4, 2, figsize=(6, 10))

# S_T_Admission
df['S_T_Admission'].value_counts().plot.pie(ax=axes[0, 0], autopct='%1.1f%%',␣
↪textprops=textprops)
axes[0, 0].set_title('S_T_Admission')

# S_T_Treatment
df['S_T_Treatment'].value_counts().plot.pie(ax=axes[0, 1], autopct='%1.1f%%',␣
↪textprops=textprops)
axes[0, 1].set_title('S_T_Treatment')

# S_T_Visits
df['S_T_Visits'].value_counts().plot.pie(ax=axes[1, 0], autopct='%1.1f%%',␣
↪textprops=textprops)
axes[1, 0].set_title('S_T_Visits')

# S_Reliability
```

```
df['S_Reliability'].value_counts().plot.pie(ax=axes[1, 1], autopct='%1.1f%%',␣
 ↪textprops=textprops)
axes[1, 1].set_title('S_Reliability')

# S_Options
df['S_Options'].value_counts().plot.pie(ax=axes[2, 0], autopct='%1.1f%%',␣
 ↪textprops=textprops)
axes[2, 0].set_title('S_Options')

# S_Hours_Treatment
df['S_Hours_Treatment'].value_counts().plot.pie(ax=axes[2, 1], autopct='%1.
 ↪1f%%', textprops=textprops)
axes[2, 1].set_title('S_Hours_Treatment')

# S_Staff
df['S_Staff'].value_counts().plot.pie(ax=axes[3, 0], autopct='%1.1f%%',␣
 ↪textprops=textprops)
axes[3, 0].set_title('S_Staff')

# S_Active_Listening
df['S_Active_Listening'].value_counts().plot.pie(ax=axes[3, 1],␣
 ↪textprops=textprops)
axes[3, 1].set_title('S_Active_Listening')
plt.tight_layout()
plt.show()

# display descriptive statistics for the survey items
df[['S_T_Admission', 'S_T_Treatment', 'S_T_Visits', 'S_Reliability',␣
 ↪'S_Options', 'S_Hours_Treatment', 'S_Staff', 'S_Active_Listening']].
 ↪describe().transpose()
```

- The pie charts and summary show similar pattern across with some survey responses having almost identical proportions. This could indicate a lack of variability in the responses, which may impact the predictive power of these variables in the regression model.

[ ]:
```
# Create a correlation matrix
#  the columns correlation matrix
cols = ['S_T_Admission', 'S_T_Treatment', 'S_T_Visits', 'S_Reliability',
        'S_Options', 'S_Hours_Treatment', 'S_Staff', 'S_Active_Listening']

corr_matrix = df[cols].corr()

# Create a heatmap of the correlation matrix
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0, vmin=-1, vmax=1)
plt.title('Correlation Matrix')
plt.tight_layout()
plt.show()
```

- The correlation matrix shows that there may indeed be correlation amongst the different survey items. This could introduce multicolinarity into the regression model.
- These pairs of items have correlation coefficients very close to zero, suggesting that there is little to no linear relationship between them. When selecting variables for a regression model, these items might be preferred as they are less likely to introduce multicollinearity issues. We will note this during the initial model building phase. And check the VIF scores to confirm.
- S_T_Admission and S_Reliability: -0.0046
- S_T_Visits and S_Reliability: -0.0063
- S_T_Admission and S_Options: -0.0037
- S_T_Treatment and S_Options: -0.01
- S_T_Visits and S_Options: -0.01

10 Bivariate Visualizations

[ ]:
```
# Bivariate Graphs with Initial_days
plt.figure(figsize=(10, 6))

#  vitD_levels vs. Initial_days
plt.subplot(2, 2, 1)
sns.regplot(data=df, x='VitD_levels', y='Initial_days',␣
↪scatter_kws={'edgecolor':'black'}, line_kws={'color':'orange'})
plt.title('VitD_levels vs. Initial_days')

#Children vs. Initial_days
plt.subplot(2, 2, 2)
sns.regplot(data=df, x='Children', y='Initial_days', scatter_kws={'edgecolor':
↪'black'}, line_kws={'color':'orange'})
plt.title('Children vs. Initial_days')

# Aage vs. Initial_days
plt.subplot(2, 2, 3)
sns.regplot(data=df, x='Age', y='Initial_days', scatter_kws={'edgecolor':
↪'black'}, line_kws={'color':'orange'})
plt.title('Age vs. Initial_days')

# income vs. Initial_days
plt.subplot(2, 2, 4)
sns.regplot(data=df, x='Income', y='Initial_days', scatter_kws={'edgecolor':
↪'black'}, line_kws={'color':'orange'})
plt.title('Income vs. Initial_days')

plt.tight_layout()
plt.show()

plt.figure(figsize=(5, 3))
sns.regplot(data=df, x='Doc_visits', y='Initial_days', scatter_kws={'edgecolor':
↪'black'}, line_kws={'color':'orange'})
plt.title('Initial_days vs. Doc_visits')
plt.show()
```

- `Vitamin D` levels and initial days don't seem to have a clear pattern, with no obvious relationship. When it comes to `children` ther is no distinct trend, suggesting the number of children doesn't linearly affect the length of hospital stay. `Age` shows a spread of data across the age range without a strong trend. For `income`, there's more variability at higher income levels, but there is no clear pattern suggesting a strong relationship. Overall, these plots suggest that individually, these variables do not have a simple linear relationship with the number of initial days spent in the hospital. However, together they might. Income might benifit from a transformation to better understand the relationship. `Doc_visits` suggest that there is no strong, straightforward relationship between the number of doctor visits and the average initial days, as increased doctor visits do not correlate with either a significant increase or decrease in the initial days. The bimodal distrubution of Initial_days may be why one sees distributions grouped above and below the lines.

[ ]:
```
# Bivariate Graphs with Initial_days
plt.figure(figsize=(8, 8))

# Marital
plt.subplot(2, 2, 1)
sns.boxplot(data=df, x='Marital', y='Initial_days', color='lightgreen')
sns.pointplot(data=df, x='Marital', y='Initial_days', color='red', estimator=np.
↪mean, errorbar=None)
plt.title('Initial_days vs Marital statuses')

# Gender
plt.subplot(2, 2, 2)
sns.boxplot(data=df, x='Gender', y='Initial_days', color='lightgreen')
sns.pointplot(data=df, x='Gender', y='Initial_days', color='red', estimator=np.
↪mean, errorbar=None)
plt.title('Initial_days vs Gender categories')

# Initial_days for Readmission status
plt.subplot(2, 2, 3)
sns.boxplot(data=df, x='ReAdmis', y='Initial_days', color='lightgreen')
sns.pointplot(data=df, x='ReAdmis', y='Initial_days', color='red', estimator=np.
↪mean, errorbar=None)
plt.title('Initial_days for Readmission status')

# Initial_days vs Area categories
plt.subplot(2, 2, 4)
```

```
sns.boxplot(data=df, x='Area', y='Initial_days', color='lightgreen')
sns.pointplot(data=df, x='Area', y='Initial_days', color='red', estimator=np.
 ↪mean, errorbar=None)
plt.title('Initial_days vs Area')

plt.tight_layout()
plt.show()

# boxplot with Initial_admin and Initial_days
plt.figure(figsize=(5, 4))
sns.boxplot(data=df, x='Initial_admin', y='Initial_days', color='lightgreen')
sns.pointplot(data=df, x='Initial_admin', y='Initial_days', color='red',␣
 ↪estimator=np.mean, errorbar=None)
plt.title('Boxplot for Initial_admin and Initial_days')
plt.show()
```

- The `Marital` statuses plot shows that seperated and single patients tend to have the highest number of days in the hospital, and that divorced and married tended to spend fewer days. The signifigance of this is unknown. The `Gender` categories show slightly higher median Initial_days for males and non binary patients and a notably lower median for females compared to males. The `Readmission` plot is very interesting. It shows a significantly higher median and and grouping of `Initial_days` for readmitted patients compared to non-readmitted patients, with several outliers showing long stays among readmitted patients. Curiously, this appears to also show two distinct groups. This is noted. Lastly, the Area plot demonstrates a slightly lower median for urban areas, suggesting hospital time is less for urban patients compared to rural and suburban patients. Interstingly, `Initial_admin` shows a higher median for elective admissions compared to emergency admissions.

- *The red lines in the boxplots show the mean values for each group. This is more about practice with visualizations than anything and to quickly compare the mean to the median. If the mean is far away from the median, could suggests that the distribution of `Initial_days` within the category is skewed, possibly indicating the presence of outliers or a non-normal distribution. However, the skewness alone does not directly tell one that it is a poor candidate for the model. Additionally, the order of the arrangement of categories on the x-axis of can influence the interpretation and direction of the mean line trend. If the categories are arranged in a certain order, the mean line might appear to trend up, down, or remain flat. So it's important to not draw those kinds of conclusions from the mean line alone.*

```
[ ]: # 4 by 2 subplot grid
fig, axes = plt.subplots(4, 2, figsize=(8, 16))

# S_T_Admission
sns.boxplot(data=df, x='S_T_Admission', y='Initial_days', ax=axes[0, 0],␣
 ↪color='lightblue')
sns.pointplot(data=df, x='S_T_Admission', y='Initial_days', ax=axes[0, 0],␣
 ↪color='brown', estimator=np.mean, errorbar=None)
axes[0, 0].set_title('S_T_Admission vs. Initial_days')
```

```
# S_T_Treatment
sns.boxplot(data=df, x='S_T_Treatment', y='Initial_days', ax=axes[0, 1],␣
 ↪color='lightblue')
sns.pointplot(data=df, x='S_T_Treatment', y='Initial_days', ax=axes[0, 1],␣
 ↪color='brown', estimator=np.mean, errorbar=None)
axes[0, 1].set_title('S_T_Treatment vs. Initial_days')

# S_T_Visits
sns.boxplot(data=df, x='S_T_Visits', y='Initial_days', ax=axes[1, 0],␣
 ↪color='lightblue')
sns.pointplot(data=df, x='S_T_Visits', y='Initial_days', ax=axes[1, 0],␣
 ↪color='brown', estimator=np.mean, errorbar=None)
axes[1, 0].set_title('S_T_Visits vs. Initial_days')

# S_Reliability
sns.boxplot(data=df, x='S_Reliability', y='Initial_days', ax=axes[1, 1],␣
 ↪color='lightblue')
sns.pointplot(data=df, x='S_Reliability', y='Initial_days', ax=axes[1, 1],␣
 ↪color='brown', estimator=np.mean, errorbar=None)
axes[1, 1].set_title('S_Reliability vs. Initial_days')

# S_Options
sns.boxplot(data=df, x='S_Options', y='Initial_days', ax=axes[2, 0],␣
 ↪color='lightblue')
sns.pointplot(data=df, x='S_Options', y='Initial_days', ax=axes[2, 0],␣
 ↪color='brown', estimator=np.mean, errorbar=None)
axes[2, 0].set_title('S_Options vs. Initial_days')

# S_Hours_Treatment
sns.boxplot(data=df, x='S_Hours_Treatment', y='Initial_days', ax=axes[2, 1],␣
 ↪color='lightblue')
sns.pointplot(data=df, x='S_Hours_Treatment', y='Initial_days', ax=axes[2, 1],␣
 ↪color='brown', estimator=np.mean, errorbar=None)
axes[2, 1].set_title('S_Hours_Treatment vs. Initial_days')

# S_Staff
sns.boxplot(data=df, x='S_Staff', y='Initial_days', ax=axes[3, 0],␣
 ↪color='lightblue')
sns.pointplot(data=df, x='S_Staff', y='Initial_days', ax=axes[3, 0],␣
 ↪color='brown', estimator=np.mean, errorbar=None)
axes[3, 0].set_title('S_Staff vs. Initial_days')

# S_Active_Listening
sns.boxplot(data=df, x='S_Active_Listening', y='Initial_days', ax=axes[3, 1],␣
 ↪color='lightblue')
```

```
sns.pointplot(data=df, x='S_Active_Listening', y='Initial_days', ax=axes[3, 1],␣
 ↪color='brown', estimator=np.mean, errorbar=None)
axes[3, 1].set_title('S_Active_Listening vs. Initial_days')

plt.tight_layout()
plt.show()
```

- Seeing these survey results in a bivariate plot with Initial_days is interesting in the variation in the median Initial_days across the different survey responses. Admittedly, I am a little unsure about how to interpret this. The initial thinking is that there is some interesting insights to gleam from this. Perhaps this suggests that the survey responses may have some predictive power in determining the length of a patient's hospital stay.
- It is interesting to see the relationship between the highest survey responses and the length of hospital stay though, where it the highest rating in most categorys is correlated with the highest and lowest hospital stay lengths. This may be insightful to hospitals in terms of patient care and satisfaction.

[ ]:
```
df = pd.read_csv('df_reduced.csv', index_col=0)
```

11 C4 Data Transformation

11.1 Reexpression of categorical variables

- Since the dataset contains several nominal categorical variables, it is essential to re-express these variables in a numerical format to include them in the regression model. This process is known as one-hot encoding, and it involves re-expressing categorical variables as binary variables, a format the regression model can use, by creating dummy variables for each category within a categorical variable. The Pandas library provides a method for performing this transformation using the `pd.get_dummies()` function. This function creates a new binary column for each category in a categorical variable,; 1 indicating the presence of that category and 0 indicating the absence. The original categorical variable is then dropped from the dataset to avoid multicollinearity issues in the regression model.
- Ordinal and binary variables *(Yes/No->1/0)* will be re-expressed as well using pythons `replace` method.
- (`replace` is apparently being depreciated: "*FutureWarning: Downcasting behavior in replace is deprecated and will be removed in a future version. To retain the old behavior, explicitly call result.infer_objects(copy=False). To opt-in to the future behavior, set pd.set_option('future.no_silent_downcasting, True)'df[binary_vars] = df[binary_vars].replace({'Yes': 1, 'No': 0})*")

[ ]:
```
# select and show values counts for binary variables to compare berfore and␣
 ↪after reexpression
binary_vars = [col for col in df.columns if df[col].isin(['Yes', 'No']).all()]
for col in binary_vars:
    print(df[col].value_counts())
```

```
[ ]: # re-expression of binary variables
     df[binary_vars] = df[binary_vars].replace({'Yes': 1, 'No': 0})

     # check the unique values for the binary variables
     for col in binary_vars:
         print(df[col].value_counts())
```

- The data dictionary states: “The (Survey) variables represent responses to an eight question survey asking customers to rate the importance of various factors/surfaces on a scale of 1 to 8 (1 most important, 8 least important)” Generally 1 is low and 8 is high in terms of importance and having 1 as most important and 8 as least important is not intuitive, and has caused this analyst confusion. Therefore, we will reverse the scale of survey variables so that 1 is the lowest and 8 is the highest in terms of importance. This will make the interpretation more intuitive in the regression analysis.

```
[ ]: # Collect values and compare before and after
     survey_vars = ['S_T_Admission', 'S_T_Treatment', 'S_T_Visits', 'S_Reliability',␣
      ↪'S_Options', 'S_Hours_Treatment', 'S_Staff', 'S_Active_Listening']
     df[survey_vars].head()
```

```
[ ]: # reverse the scale of survey variables so that 1 is the lowest and 8 is the␣
      ↪highest in terms of importance. and after the change.
     df[survey_vars] = df[survey_vars].replace({1: 8, 2: 7, 3: 6, 4: 5, 5: 4, 6: 3,␣
      ↪7: 2, 8: 1})
     df[survey_vars].head()
```

```
[ ]: # reexpress `complication_risk` from categorical variable to numerical variable␣
      ↪where 1 is the lowest and 3 is the highest in terms of risk.
     # Check values before and after
     print(df['Complication_risk'].value_counts())
     df['Complication_risk'] = df['Complication_risk'].replace({'Low': 1, 'Medium':␣
      ↪2, 'High': 3}).astype(int)
     df['Complication_risk'].value_counts()
```

```
[ ]: #to csv to save progress so far.
     df.to_csv('df_for_one_hot.csv', index='CaseOrder')
```

```
[ ]: #read the csv
     df = pd.read_csv('df_for_one_hot.csv', index_col=0)
     df.head().transpose()
```

- To handle nominal variables (categorical variables with no inherent order) in a regression model, one-hot encoding is often used. This transforms each unique category of a variable into a separate binary variable. Each new binary variable represents the presence (1) or absence (0) of the category for a data point. (Middleton, 2022)

- To avoid introducing multicollinearity, it’s common practice to drop one of the binary variables from each encoded category. Which will be done with the optional argument `drop_first=True` in the `pd.get_dummies` method.

```
[ ]: # Using get_dummies to convert nominal variables to 1 and 0 for one-hot␣
      ↪encoding and drop the first column to avoid multicollinearity.
     nominal_vars = ['Area', 'Marital', 'Gender', 'Initial_admin']
     df_encoded = pd.get_dummies(df, columns=nominal_vars, dtype=int,␣
      ↪drop_first=True)
```

```
[ ]: # Show the head of the encoded DataFrame
     pd.set_option('display.max_columns', None)
     df_encoded
```

- Noting the shape to be sure extra columns drop: 38 columns

```
[ ]: #FINAL CLEAN TRANSFORMED CSV
     df_encoded.to_csv('medical_transformed.csv', index='CaseOrder')
```

- Import Transformed Data for Initial model

```
[ ]: df = pd.read_csv('medical_transformed.csv', index_col=0)
     df.head()
```

```
[ ]: # create a histogram of the initial_days column
     plt.figure(figsize=(4, 3))
     plt.hist(df['Initial_days'], bins=50)
     plt.xlabel('Initial_days')
     plt.show()
```

---

12 Part IV: Model Comparison and Analysis

12.1 D. Compare an initial and a reduced linear regression model by doing the following:

12.1.1 D1. Construct an initial multiple linear regression model from all independent variables that were identified in part C2:

The processes and code below were informed by several source mentioned in the refrence section. (Sewell, 2024), (UnfoldDataScience YouTube, 2023), (Stack Overflow, 2020), (GeeksforGeeks, 2023), (Indhumathy Chelliah, 2021)

```
[ ]: # muultiple regression model using df and ols and 'Initial_days' as the␣
      ↪dependent variable and all other variables in the dataset as independent␣
      ↪variables
     X = df.drop('Initial_days', axis=1)
     Y = df['Initial_days']
     X = sm.add_constant(X)
     model = sm.OLS(Y, X).fit()
     predictions = model.predict(X)
```

```
model_summary = model.summary()
model_summary
```

13 Initial Regression Model based on all predictors

- **($\hat{y}$) = 19.4602 + 0.0401(Children) + 0.0035(Age) - 3.442e-06(Income) + 46.4505(ReAdmis) - 0.0775(VitD_levels) - 0.1714(Doc_visits) + 0.2924(vitD_supp) - 0.4475(HighBlood) - 0.2008(Stroke) - 0.3944(Complication_risk) - 0.2090(Overweight) + 0.6649(Arthritis) + 0.0132(Diabetes) - 0.3959(Hyperlipidemia) + 0.3505(BackPain) + 0.5303(Anxiety) + 0.4092(Allergic_rhinitis) + 0.4223(Reflux_esophagitis) + 0.0406(Asthma) + 0.4003(S_T_Admission) + 0.1342(S_T_Treatment) - 0.1296(S_T_Visits) + 0.3911(S_Reliability) + 0.0093(S_Options) - 0.2056(S_Hours_Treatment) - 0.2466(S_Staff) - 0.1981(S_Active_Listening) + 0.1602(Area_Suburban) + 0.3731(Area_Urban) - 0.0263(Marital_Married) + 0.4302(Marital_Never Married) + 0.7953(Marital_Separated) + 0.2762(Marital_Widowed) - 0.0963(Gender_Male) - 0.2836(Gender_Nonbinary) - 1.6011(Initial_admin_Emergency Admission) - 0.2463(Initial_admin_Observation Admission)**

13.0.1 D2. Justify a statistically based feature selection procedure or a model evaluation metric to reduce the initial model in a way that aligns with the research question:

- A `backwards selection` method will be used to reduce the initial model. To justify the feature selection, model summary statistics will be analyzed, certain assumptions will be checked, and visualizations will be created and analyzed.

13.0.2 Note: The following requirements from `Part E` of the performance assessment will be demonstrated in the multiple cells below, but not necessarly in the exact order of the PA.

-

  #### 13.0.3 *E. Analyze the data set using your reduced linear regression model by doing the following:*

- *E1. Explain your data analysis process by comparing the initial multiple linear regression model and reduced linear regression model, including the following element:** • a model evaluation metric*

- *E2. Provide the output and all calculations of the analysis you performed, including the following elements for your reduced linear regression model:** • a residual plot • the model's residual standard error*

```
[ ]: fig, axes = plt.subplots(1, 3, figsize=(15, 5))

     #  actual vs predicted initial days
```

```
sns.regplot(x=predictions, y=Y, fit_reg=True, line_kws={'color':'orange'},␣
 ↪ax=axes[0])
axes[0].set_xlabel('Predicted Initial Days')
axes[0].set_ylabel('Actual Initial Days')
axes[0].set_title('Actual vs Predicted Initial Days')

#  histogram of actual values
axes[1].hist(Y, bins=100)
axes[1].set_xlabel('Initial Days')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Histogram of Actual Initial Days')

# histogram of predicted values
axes[2].hist(predictions, bins=100)
axes[2].set_xlabel('Predicted Initial Days')
axes[2].set_ylabel('Frequency')
axes[2].set_title('Histogram of Predicted Initial Days')

plt.tight_layout()
plt.show()
```

- The plot of `Actual vs Predicted days` seems to miss much of the data in the middle, predicting at lower and higher values. The `actual Initial Days` histogram again shows it's bimodal distribution, suggesting two distinct groups or patterns within the data. The `predicted Initial Days` histogram clearly shows a large range of missing values in the middle, and is heavily concentrated at the low and high ends of the range.

[ ]:
```
# calculate RSE
mse = model.scale
# Calculate RSE
rse = np.sqrt(mse)
print("Residual Standard Error (RSE):", rse)
```

Rse calculation: (Stack Overflow 2023)

13.1 Initial Model Fit:

- The R-squared is 0.726, suggesting that approximately 72.6% of the variability in `Initial_days` may be explained by the model, which in combination with the almost identical Adj. R-squared of 0.725 indicating a good fit with initial model.
- The F-statistic is 714.14 with a Prob (F-statistic) of 0.00, *suggesting* that the model is statistically significant overall.
- The AIC 8.090e+04 and BIC 8.117e+04 are very similar, suggesting suggests that both are close in their evaluation of model complexity. These will be re-examined in the reduced model to see if they are lowered.
- Residual Standard Error calculation (RSE): 13.79 `Initial_days` ranges from 1 to 72 days. An RSE of 13.79 days represents over 19% (13.792 / 72 * 100) of the total range. This suggests that, on average, the model's predictions for length of stay can deviate from the

actual values by up to 13.792 days. This seems significant in this context and indicates room for improvement.

Variables. Some predictor variables have high t-values and low p-values (P>|t|), sometimes indicating that they are statistically significant. Of note is the `ReAdmis` feature, it has a very low p-value and a highly significant coefficient (46.4505), suggesting a strong association with `Initial_days` However, variables such as Children, Age, Income, and others have high p-values, indicating that they might not be significant predictors of `Initial_days` in the presence of other variables. The const coefficient (y-intercept) is 19.5835, which represents the expected value of `Initial_days` when all other predictors are at zero. It is also important to note the values of the coefficients for the predictors. Larger values can suggest a more important role for the predictor in the model. For example, the coefficient for `ReAdmis` is 46.4505, which means that for every unit increase in `ReAdmis`, the expected value of `Initial_days` increases by 46.4505 units, holding all other predictors constant. This is a large coefficient compared to others, indicating a strong relationship between `ReAdmis` and `Initial_days`.

- Overall, even though there are some summary statistics points to a reliable model, we should be skeptical with the warning about potential multicollinearity, and the plot of the actual vs predicted values not accounting for much of the data. There are several things we should check.

Issues: The note on multicollinearity "[2] The condition number is large, 8.89e+05. This might indicate that there are strong multicollinearity or other numerical problems." indicates that there might be high correlation between some predictors. This needs to be investigated. Checking the residuals and Variance inflation factor (VIF) analysis can help to identify and highly correlated predictors.

[ ]:
```python
residuals = model.resid
# Plot the residuals
fig, axes = plt.subplots(1, 3, figsize=(10, 3))

# reesiduals Dist
sns.histplot(residuals, bins=30, ax=axes[0])
axes[0].set_xlabel('Residuals')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Residuals Distribution')

# mean and median lines to the histogram
mean_residuals = residuals.mean()
median_residuals = residuals.median()
axes[0].axvline(x=mean_residuals, color='green', linestyle='--', label='Mean')
axes[0].axvline(x=median_residuals, color='blue', linestyle='--',␣
  ↪label='Median')
axes[0].legend()

# QQ plot of residuals
sm.qqplot(residuals, line='s', ax=axes[1])
axes[1].set_xlabel("Theoretical Quantiles")
axes[1].set_ylabel("Sample Quantiles")
```

```
axes[1].set_title("QQ plot of residuals")

plt.axhline(y=0, color='red', linestyle='--')
sns.scatterplot(x=predictions, y=residuals, ax=axes[2])
axes[2].set_xlabel("Predicted values")
axes[2].set_ylabel("Residuals")
axes[2].set_title("Residuals vs. predicted values")

# mean and median lines
mean_residuals = residuals.mean()
median_residuals = residuals.median()
axes[2].axhline(y=mean_residuals, color='green', linestyle='--', label='Mean')
axes[2].axhline(y=median_residuals, color='blue', linestyle='--',␣
 ↪label='Median')
axes[2].legend()

plt.tight_layout()
plt.show()
```

- Ideally, one would like to see a normal distribution centered around zero for the residual distribution, but this **histogram** indicates a slightly bimodal distribution that is skewed to the right, with a tail that grows slightly positive values. Hoever, the mean is in fact around 0, which presents and interesting challenge.
- The **Q-Q plot** shows that the residuals are not normally distributed, as the points do not fall along the straight line. This plot required research to interpret as I was not familiar with it. From S. Kross as seankross.com: "*The points in Q-Q plot then cross below the blue line indicating that the actual quantiles that are close to zero are farther from zero than they should be theoretically. At the center of the theoretical distribution there are no data in the actual dataset, and therefore there is no point in the Q-Q plot at (0, 0). The upper half of the Q-Q plot is a reflection across X and Y of the bottom half.*" (Kross, 2016) The author also suggest this is the results of a residual distribution that is similar to the one I have. Additionally, the center of the theoretical distribution indeed did not have any data points as mentioned above.
- According to The residuals (errors) should be scattered randomly above and below the zero line across the entire range of predicted value and There should be no discernible pattern in the scatter plot of residuals. This is not the case in my scatter plot of residuals. Indicating that the model is not a good fit for the data.
- VIF analysis will be used to identify highly correlated predictors.

```
[ ]: # perform VIF analysis to check for multicollinearity
X = add_constant(X)
vif = pd.DataFrame()
vif["variables"] = X.columns
vif["VIF"] = [variance_inflation_factor(X, i) for i in range(X.shape[1])]
vif.sort_values(by='VIF', ascending=False)
```

From the VIF analysis, most variables have VIF values well below 5, indicating no significant multicollinearity among them, which is surprising given the message from the model summary: *"[2] The condition number is large, 8.89e+05. This might indicate that there are strong multicollinearity or other numerical problems."* The highest VIF values observed for `S_T_Admission`, `S_T_Treatment`, and `marital` status categories, but even these do not exceed the threshold of 5, suggesting moderate correlation at most. (Stack Exchange, 2012)

Given the generally low VIF values, the summary statistics mentioned above the suggest a good model fit, the residuals and Q-Q plots, and the actual and predicted values, something less obvoius must be wrong with the initial model. It is also important to keep in mind the bimodal distribution of the actual `Initial_days` values, which may be contributing to the model's poor performance due to the violation of the assumption of normality.

One option is to try to transform the `Initial_days` variable to make it more normally distributed. This could involve taking the log of the variable, or splitting the data into two groups based on the bimodal distribution and creating separate models for each group.(Bradley, 2023) However, this would likely be complex at this point in the process and may not be necessary if a reduced model can be created that performs better than the initial model. Therefore, before that is attempted, I will employ what *domain knowledge* I have regarding the variable and use a *correlation matrix* to help identify pairwise relationships between the independent variable. In this context, perhaps a better model can be created by removing some of the predictors that are not significantly associated with `Initial_days` and may be contributing to the model's poor performance.

```
[ ]: # assign independent variables
     corr_matrix = df.drop('Initial_days', axis=1).corr()

     plt.figure(figsize=(20, 12))

     # correlation matrix with  values and adjusted font size
     sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0,␣
      ↪annot_kws={"size": 8}, vmin=-1, vmax=1)

     plt.title('Correlation Matrix of Independent Variables')
     plt.show()
```

- The correlation matrix shows that most of the variables that contain any correlation wit each other are the survey items, marital status, area, and initial admission. These would be interesting to explore further but in the context of this analyst, are not as useful as the other predictors which are largely health and biological factors and importantly, readmission status. The survey items are likely to be highly correlated with each other, and subjective feedback worthy of their own separate analyses, but here may be getting in the way of the model.Marital status, in my personal experience, has never been a factor health care providers consider except when contacting family members. I feel that these are good candidates for removal from the model.

```
[ ]: df = df.drop(['S_T_Admission', 'S_T_Treatment', 'Marital_Widowed',␣
      ↪'Marital_Married', 'Marital_Never Married', 'Marital_Separated',␣
      ↪'S_T_Visits', 'S_Hours_Treatment', 'S_Reliability', 'S_Staff','S_Options',␣
      ↪'Area_Urban','Initial_admin_Observation Admission', 'S_Active_Listening',␣
      ↪'Area_Suburban','Initial_admin_Emergency Admission'], axis=1)
```

```
df.head().transpose()
```

[ ]:
```
df.to_csv('medical_transformed_drop1.csv', index='CaseOrder')
```

[ ]:
```
# read in the new csv file
df = pd.read_csv('medical_transformed_drop1.csv', index_col=0)
```

[ ]:
```
X = df.drop(['Initial_days'], axis=1)
Y = df['Initial_days']
X = sm.add_constant(X)
model_2 = sm.OLS(Y, X).fit()
predictions = model_2.predict(X)
residuals_2 = model_2.resid
model_summary_2 = model_2.summary()
model_summary_2
```

[ ]:
```
# calculate RSE
mse = model_2.scale
# Calculate RSE
rse = np.sqrt(mse)
print("Residual Standard Error (RSE):", rse)
```

- The R-squared went from 0.726 to 0.725
- Adj. R-squared went from 0.725 to 0.724
- The F-statistic went from 714.14 with a Prob (F-statistic) of 0.0 to 1314 with a Prob (F-statistic) of 0.0. The f-statistic increase suggests that the model is a better fit than the previous model.
- The AIC and BIC went from 8.090e+04 and 8.117e+04 to AIC 8.092e+04 and BIC 8.107e+04 are very similar to the previous model.
- Residual Standard Error calculation (RSE): 13.79 to 13.8, almost unchanged
- The condition number went from 8.88e+05 to 5.64e+05 which is a significant improvement, suggesting that the multicollinearity is less of an issue in this model. But still the condition number is high, so multicollinearity may still be a problem.

[ ]:
```
# Calculate the residuals_2
residuals_2_2 = model_2.resid

# Plot the residuals_2
fig, axes = plt.subplots(1, 3, figsize=(10, 3))

# iduals_2 Distribution
sns.histplot(residuals_2, bins=30, ax=axes[0])
axes[0].set_xlabel('Residuals_2residuals_2')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Residuals_2residuals_2 Distribution')

#mean and median lines
mean_residuals_2 = residuals_2.mean()
```

```
median_residuals_2 = residuals_2.median()
axes[0].axvline(x=mean_residuals_2, color='green', linestyle='--', label='Mean')
axes[0].axvline(x=median_residuals_2, color='blue', linestyle='--',␣
↪label='Median')
axes[0].legend()

# QQ plot
sm.qqplot(residuals_2, line='s', ax=axes[1])
axes[1].set_xlabel("Theoretical Quantiles")
axes[1].set_ylabel("Sample Quantiles")
axes[1].set_title("QQ plot of residuals_2")

plt.axhline(y=0, color='red', linestyle='--')
sns.scatterplot(x=predictions, y=residuals_2, ax=axes[2])
axes[2].set_xlabel("Predicted values")
axes[2].set_ylabel("Residuals_2")
axes[2].set_title("Residuals_2 vs. predicted values")

# Add mean and median lines
mean_residuals_2 = residuals_2.mean()
median_residuals_2 = residuals_2.median()
axes[2].axhline(y=mean_residuals_2, color='green', linestyle='--', label='Mean')
axes[2].axhline(y=median_residuals_2, color='blue', linestyle='--',␣
↪label='Median')
axes[2].legend()

plt.tight_layout()
plt.show()
```

- Not much has changed in our residual plots. Although our model is slightly improved according to the F-statistic, the R-squared and the condition number, the residual plots are still showing some patterns that suggest that the model is not capturing all the information in the data.
- Let's check VIF values for each of the predictors again to see if we gain ant new insights.

[ ]:
```
# perform VIF analysis to check for multicollinearity
X = add_constant(X)
vif = pd.DataFrame()
vif["variables"] = X.columns
vif["VIF"] = [variance_inflation_factor(X, i) for i in range(X.shape[1])]
vif.sort_values(by='VIF', ascending=False)
```

- The VIF values for the predictors are all essentially 1.
- Given this information, re-examining the p-values of the predictors and their coefficients, we can eliminate the predictors with the highest p-values and lower coefficients to see if that improves the model. Here we have to be careful, because predictors with high p-values may still be important for the model. Examining the p-values and the coefficients together,

- After reviewing p-values and coefficients, I am choosing to remove based on coefficient as it is associated with higher p-values.
- Eliminate those variables with `coefficients less than an absolute value of 0.4`.

```
[ ]: # statistically significant variables
     significant_vars = model_2.params[model_2.params.abs() > 0.4].index.tolist()

     # Remove 'const' from the list
     if 'const' in significant_vars:
         significant_vars.remove('const')

     print('Significant variables:', significant_vars)
```

```
[ ]: # Create a reduced model with only significant variables above
     X_reduced = X[significant_vars]

     # Fit the OLS model with reduced variables
     model_reduced = sm.OLS(Y, sm.add_constant(X_reduced)).fit()

     # Print the summary of the reduced model
     model_reduced.summary()
```

```
[ ]: # calculate RSE
     mse = model_reduced.scale
     # Calculate RSE
     rse = np.sqrt(mse)
     print("Residual Standard Error (RSE):", rse)
```

```
[ ]: # new residual check and assign
     residuals_reduced = model_reduced.resid
     residuals_reduced
```

```
[ ]: # Plot the residuals_reduced
     fig, axes = plt.subplots(1, 3, figsize=(10, 3))

     # residuals_reduced Distribution
     sns.histplot(residuals_reduced, bins=30, ax=axes[0])
     axes[0].set_xlabel('residuals_reduced')
     axes[0].set_ylabel('Frequency')
     axes[0].set_title('residuals_reduced Distribution')

     #  mean and median lines  the histogram
     mean_residuals_reduced = residuals_reduced.mean()
     median_residuals_reduced = residuals_reduced.median()
     axes[0].axvline(x=mean_residuals_reduced, color='green', linestyle='--',␣
      ↪label='Mean')
```

```
axes[0].axvline(x=median_residuals_reduced, color='blue', linestyle='--',␣
 ↪label='Median')
axes[0].legend()

# QQ plot of residuals_reduced
sm.qqplot(residuals_reduced, line='s', ax=axes[1])
axes[1].set_xlabel("Theoretical Quantiles")
axes[1].set_ylabel("Sample Quantiles")
axes[1].set_title("QQ plot of residuals_reduced")

plt.axhline(y=0, color='red', linestyle='--')
sns.scatterplot(x=predictions, y=residuals_reduced, ax=axes[2])
axes[2].set_xlabel("Predicted values")
axes[2].set_ylabel("residuals_reduced")
axes[2].set_title("residuals_reduced vs. predicted values")

# Add mean and median
mean_residuals_reduced = residuals_reduced.mean()
median_residuals_reduced = residuals_reduced.median()
axes[2].axhline(y=mean_residuals_reduced, color='green', linestyle='--',␣
 ↪label='Mean')
axes[2].axhline(y=median_residuals_reduced, color='blue', linestyle='--',␣
 ↪label='Median')
axes[2].legend()

plt.tight_layout()
plt.show()
```

Part V: Data Summary and Implications

F. Summarize your findings and assumptions by doing the following:

1. Discuss the results of your data analysis, including the following elements:
    - a regression equation for the reduced model
    - an interpretation of the coefficients of the reduced model
    - the statistical and practical significance of the reduced model
    - the limitations of the data analysis
2. Recommend a course of action based on your results.

13.1.1 Reduced model:

$(\hat{y})$ = 17.3266 + 46.4413(ReAdmis) - 0.4551(HighBlood) + 0.6742(Arthritis) - 0.4176(Hyperlipidemia) + 0.5454(Anxiety)

MODEL Statistical Significance SUMMARY: - The**R-squared** went from 0.726 to 0.725 to `0.724` - **Adj. R-squared** went from 0.725 to 0.724 to `0.724` as well. - The **F-statistic** went from 714.14 to 1314 to `5252` all with a Prob (F-statistic) of 0.0. The f-statistic increase suggests that the model is a much better fit than the previous model. - The **AIC** and **BIC** went from 8.090e+04 and 8.117e+04 to AIC 8.092e+04 and BIC 8.107e+04 to `AIC 8.090e+04` and `BIC 8.095e+04`, very close to each other. - **Residual Standard Error** calculation (RSE): 13.79 to remain at `13.8` no matter the model change. - The *condition number* went from 8.88e+05 to 5.64e+05 to `3.63` suggesting that the multicollinearity is less of an issue in this model. Likely helped by the remove of features. - a **Durbin-Watson** statistic of 1.269 suggests that there is some positive autocorrelation in the model's residuals. This could potentially be a problem because the assumption of independence of errors may be violated.

(ŷ) = 17.3266 + 46.4413(ReAdmis) - 0.4551(HighBlood) + 0.6742(Arthritis) - 0.4176(Hyperlipidemia) + 0.5454(Anxiety)

> **Interpretation of the coefficients**: *(Note, this is a very basic interpretation of what this formula tells us, this does not mean what the model is telling us is actually accurate or useful.)*

(ŷ) represents the predicted value of the dependent variable, which in this case is the *initial* number of days spent in the hospital.

- The intercept (17.3266) is the value of (ŷ) when all independent variables are zero. In other words, if a patient has not been `ReAdmitted`, has no `HighBlood` pressure, no `Arthritis`, no `Hyperlipidemia`, and no `Anxiety`, the predicted number of`Initial_days` in the hospital would be approximately `17  days`
- The coefficients for each explanatory variable indicate the change in the predicted value of (ŷ) for a *one-unit* increase in that variable, when alll other variables remain constant.

Relationships between the independent variables and the predicted number of `Initial_days` in the hospital are such:

- – A readmitted patient (`ReAdmis`) is associated with a longer hospital stay.
- – High blood pressure (`HighBlood`) is associated with a shorter hospital stay.
- – Arthritis (`Arthritis`) is associated with a longer hospital stay.
- – Hyperlipidemia (`Hyperlipidemia`) is associated with a shorter hospital stay.
- – Anxiety (`Anxiety`) is associated with a longer hospital stay.

The magnitudes of the coefficients also indicate the relative strength of each variable's association with the predicted outcome. The coefficient for ReAdmis (46.4413) is the largest, suggesting that a history of readmission has the strongest impact on the length of hospital stay among the variables included in the model. However, there is a major problem that the high coefficient and low P-value for `ReAdmis` admittedly mislead me on.

We have a model that's designed to predict how many days a patient might initially spend in the hospital, and it uses various health conditions as input factors. However, it has a significant limitation in its design and the way I used one of these factors, namely the patient's readmission status. In a real-world context, according to the data dictionary, the patient's **re**admission can only occurs after they have had and Initial stay, making readmission a *future event compared to their initial stay.* In this model, `ReAdmission` status is currently being treated as a factor that can predict the initial hospital stay. But logically, this isn't possible because one can't use information from the future to predict the pas This a fundamental issue with how the prediction model was constructed and a reminder that models aren't just about variables and coefficients, but having a fundamental understanding of the data and its context in in the real world.

Given the results and limitations of the model, my recommended course of action would include a few key steps. At this point, I wouldn't recommend using this model in its current state. If this were a real-world scenario, I would consider myself to be in the exploratory phase of the project. Taking the information we've gathered, I suggest re-evaluating both the explanatory and target variables.

One issue that stands out is the bimodal distribution of the target variable, initial days. During this project, I learned that bimodal distributions can represent two distinct populations within the data set. To address this, I recommend possibly splitting the data and performing appropriate transformations to try and make the distributions of each population more normal.

Moving forward, it's crucial to better understand all the variables. Consulting with domain experts and getting additional data analytics expertise, particularly when it comes to selecting and transforming potential explanatory variables, would be highly beneficial.

I want to highlight something interesting I noticed while creating the visualizations, although I'm not entirely sure what it means yet. If we look at the box plot comparing the `Initial_days` with the `rReAdmis` variable, it forms a pattern that stood out in the visualizations from part C. Interestingly, the scatterplot of the model's predicted values against the actual values has a strikingly similar pattern to the boxplots I mentioned. I believe there's something worth exploring here.

The next project involves logistical regression, and I'm curious to investigate whether it would be useful to explore these variables in the opposite context. Perhaps the initial days in the hospital can help predict readmission status. This is the direction I plan to pursue in my next course of investigation.

```
[ ]: fig, axes = plt.subplots(1, 2, figsize=(6, 3))

     # Actual vs Predicted Initial Days
     sns.regplot(x=predictions, y=Y, fit_reg=True, line_kws={'color':'orange'},␣
      ↪ax=axes[0])
     axes[0].set_xlabel('Predicted Initial Days')
     axes[0].set_ylabel('Actual Initial Days')
     axes[0].set_title('Actual vs Predicted Initial Days')

     # Boxplot for ReAdmis and Initial_days
     sns.boxplot(data=df, x='ReAdmis', y='Initial_days', color='lightgreen',␣
      ↪ax=axes[1])
     sns.pointplot(data=df, x='ReAdmis', y='Initial_days', color='red', estimator=np.
      ↪mean, errorbar=None, ax=axes[1])
     axes[1].set_title('Boxplot for ReAdmis and Initial_days')

     plt.tight_layout()
     plt.show()
```

14 Conclusion

This project was a remarkable learning experience, the instructors indicated that the project would force tough decisions, highlighting the absence of standout models and the essential nature of hard choices. As I reached the project's conclusion, I recognized possible mistakes and oversights along the way. Despite the urge to correct these errors, I opted to keep them, valuing the learning process over the creation of a flawless model as well as the reality of deadlines. This project provides a blueprint of my thinking; a reminder of overlooked aspects crucial for future analysis projects.

14.0.1 References

- Datacamp. (2023, December 12). D207 - Exploratory Data Analysis. Retrieved from https://app.datacamp.com/learn/custom-tracks/custom-d207-exploratory-data-analysis
- Middleton, K. (2022, November). Getting started with D208 Part I [Webinar]. Western Governors University. Retrieved March 2024, from https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=15e09c73-c5aa-439d-852f-af47001b8970
- Eyre, I. (2024, March). Python Seaborn: A Guide to Using Seaborn for Data Visualization. Real Python. Retrieved March 2024, from https://realpython.com/python-seaborn/
- Statology. (n.d.). *The Five Assumptions of Multiple Linear Regression.* Statology. Retrieved March 10, 2024, from www.statology.org/multiple-linear-regression-assumptions/
- GeeksforGeeks. (n.d.). Interpreting the results of linear regression using OLS summary. Retrieved March 24, 2024, from https://www.geeksforgeeks.org/interpreting-the-results-of-linear-regression-using-ols-summary/
- Indhumathy Chelliah. (2021, August 3). Everything to know about residuals in linear regression. Retrieved March 24, 2024, from https://indhumathychelliah.com/2021/08/03/everything-to-know-about-residuals-in-linear-regression/
- Kross, S. (2016, February 29). A Q-Q Plot Dissection Kit. Retrieved March 24, 2024, from https://seankross.com/2016/02/29/A-Q-Q-Plot-Dissection-Kit.html
- Bradley, C. (2023). How do you transform a skewed bimodal data set into a normal distribution? Retrieved March 24, 2024, from https://www.kaggle.com/code/chrisbradley/tab-playground-predicting-bimodal-distribution
- Manl, F. (n.d.). Intermediate Regression with statsmodels in Python. Retrieved March 24, 2024, from https://github.com/FraManl/DataCamp/blob/main/Intermediate%20Regression%20with%20sta
- Python Graph Gallery. (n.d.). Retrieved March 24, 2024, from https://python-graph-gallery.com/
- ResearchGate. (n.d.). Normal probability plots. Retrieved March 24, 2024, from https://www.researchgate.net/figure/Normal-pro-bability-plots-a-ideal-b-heavy-tailed-distri-bution-c-light-tailed_fig1_262663278
- Statology. (n.d.). Multiple Linear Regression Assumptions. Retrieved March 24, 2024, from https://www.statology.org/multiple-linear-regression-assumptions/
- Stack Exchange. (n.d.). Multicollinearity: When individual regressions are significant but VIFs are low. Retrieved March 24, 2024, from https://stats.stackexchange.com/questions/24464/multicollinearity-when-individual-regressions-are-significant-but-vifs-are-low

- Stack Overflow. (2020, August 10). Residual standard error of a regression in Python. Retrieved March 24, 2024, from https://stackoverflow.com/questions/63333999/residual-standard-error-of-a-regression-in-python
- Western Governors University. (n.d.). R or Python? Retrieved March 24, 2024, from https://www.wgu.edu/online-it-degrees/programming-languages/r-or-python.html
- @UnfoldDataScience YouTube. (n.d.). Checking MLR Assumptions. Retrieved March 24, 2024, from https://www.youtube.com/watch?v=_XAurJJQ7jw
- Sewell, W. (2024). [PowerPoint slides on model tuning]. Western Governors University. Retrieved March 24, 2024, from https://westerngovernorsuniversity-my.sharepoint.com/:p:/g/personal/william_sewell_wgu_edu/ERPQ0YpiQktOl-7YyAVnfLMBR5qeBh2cSv61VaJqe_aHKg?e=FjPhPz

Software References

The following software packages were used in this project:

- **pandas** pandas is a Python library providing data structures and data analysis tools.
- **numpy** NumPy is a Python library for scientific computing.
- **matplotlib** Matplotlib is a Python library for creating static, animated, and interactive visualizations.
- **seaborn** Seaborn is a Python library for statistical data visualization built on top of matplotlib.
- **statsmodels** Statsmodels is a Python library for statistical modeling and econometrics.
- **Scikit-learn** Scikit-learn is a free machine learning library for Python.

- Python Software Foundation. (2023). Python Language Reference, version 3.9.6. Retrieved from https://www.python.org
- Scikit-Learn: Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830. Retrieved from https://scikit-learn.org
- Jinja2: Ronacher, A. (2023). Jinja2: The modern and designer-friendly templating engine for Python. Retrieved from https://jinja.palletsprojects.com/
- Matplotlib: Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90-95. Retrieved from https://matplotlib.org
- Statsmodels: Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference* (Vol. 57, p. 61). Retrieved from https://www.statsmodels.org
- Pandas: McKinney, W. (2010). Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference* (pp. 51-56). Retrieved from https://pandas.pydata.org
- NumPy: Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature, 585*(7825), 357-362. Retrieved from https://numpy.org

- Seaborn: Waskom, M. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software, 6*(60), 3021. Retrieved from https://seaborn.pydata.org

```
[ ]: # manage memory by using gc.collect() to clear memory
     import gc
     gc.collect()
```

```
[ ]: 0
```