D208 – Predictive Modeling - Task 1

WGU M.S. Data Analytics

Lyssa Kline

September 9, 2024

A, Research Question

A1, Question

The research question in scope for this analysis is, "What factors greatly influence the total amount charged to a patient?" Understanding the factors that influence the total amount charged to a patient can help medical organizations manage and potentially reduce costs. This insight could lead to more efficient allocation of resources as well as better financial planning and greater transparency to patients regarding billing. Analyzing the factors affecting charges could help identify areas where costs are disproportionately high, which could lead to targeted efforts to streamline processes and improve the quality of care to patients.

A2, Analysis Goals

The goal of the analysis being performed is to determine which variables have the most significant impact on the total amount charged to a patient. Analyzing the magnitude of each factor will measure the impact each factor has in influencing total charges. The primary goal of this analysis will be to provide actionable insights that can be used to implement cost-saving measures, improve quality of care, and optimize the billing process.

B, Method Justification

B1, Linear Regression Model Assumptions

To ensure the validity of a multiple linear regression model, some assumptions need to be met. The assumptions are as follows:

- The first assumption is the relationship between the dependent variable and independent variable should be linear, which would result in proportional changes in the dependent variable.
- The second assumption is the observations in the dataset should be independent of each other, meaning the value of one observation should not be influenced by the value of another observation.
- The third assumption is the errors of the model should have constant variance at all levels of the independent variables (or homoscedasticity), meaning the spread of the residuals should be consistent across all independent variables.
- The last assumption would be that the residuals of the model should have normality, meaning a consistent spread, and be normally distributed.

These assumptions should be followed to ensure an adequate data analysis is performed providing useful insights into the data.

B2, Tool Benefits

Python was the chosen tool for this analysis due to experience, business use case, and its extensive libraries and robust tools. Python has a rich ecosystem of libraries that supports all phases of analysis making Python a powerful choice for handling complex analytical tasks. Python

scripts can automate repetitive tasks and can make an analysis reproducible, which is beneficial when building models.

Packages used in this report allow a wide range of analyses to be done. These packages are commonly used and allow various statistical analysis and reduction techniques to be performed on the selected data.

```
# Importing all packages used in this workflow.
import pandas as pd
import numpy as np
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
import statsmodels.api as sm
from sklearn.linear_model import Lasso
from scipy.stats import chi2_contingency, ttest_ind
from sklearn.feature_selection import RFE
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
```

B3, Analysis Technique

The research question chosen under A1 seeks to understand the relationship between the total amount charged (dependent variable) and various potential influencing factors (independent variables). The dependent variable in scope of TotalCharge is a numerical continuous value containing the total charged to patients for the services they received while admitted to the hospital. For this analysis, multiple linear regression is the appropriate technique due to a variety of reasons.

The use of a numerical continuous value as the dependent variable in a linear regression model offers several benefits including the support it provides to a residual analysis, accurate predictions, better trend and pattern analysis, and model flexibility.

Multiple Linear Regression modeling techniques can help quantify the relationship between the independent and dependent variables allowing for a clear understanding of which factors have the most significant impact on total charges to patients. This modeling technique can be used to predict the dependent variable, which can be valuable to medical organizations to forecast change. Additionally, a linear regression model helps to provide a straightforward measure of impact on each predictor variable, making it accessible for stakeholders to understand the results of the analysis.

C, Data Preparation

Cleaning data for analysis is the process of identifying and correcting errors in a dataset. Cleaning data helps to ensure accuracy, completeness, and consistency. To ensure an adequate linear analysis can be performed, first, the data must be cleaned to find any duplicates, missing values, outliers, and categorical variables.

C1, Data Cleaning Steps, and Goals

The primary goal of data cleaning is to improve the quality of the data by detecting and correcting errors, inconsistencies, or missing values in a dataset. High-quality, accurate, and consistent data are essential for meaningful analysis, decision-making, and building models.

The data for this course is described as 'clean' however reviewing the data for cleanliness before using it for the model is key to developing a good model. To ensure adequate cleaning techniques were followed, I used the same cleaning steps as was used in D206. The data cleaning process consisted of identifying and cleaning duplicates, missing values, outliers, and the re-expression of categorical variables. The rubric was a useful tool to assist in mitigating data cleanliness issues.

To do this the data will be assessed for all inconsistencies. the data can be reviewed for duplicates, missing values, nulls, outliers, data types, and unique values. In addition to identifying where the inconsistencies are, we can determine statistics variables, assess the summary statistics, determine cardinality, and calculate the VIF for each variable. The steps used to determine these statistics are performed under the following C2-C4 sections below.

C2, Statistical Variables

To answer the research question in the scope of "What variables influence the total amount charged to a patient", the dependent variable would be 'TotalCharge' and the independent variables selected include key patient health information that may influence the total amount charged. After the cleaning steps performed under C1 above, there were 14 independent variables selected for further analysis.

Independent Variable chosen for analysis:

- TotalCharge – Data Type: Float64, Continuous Variable – Total amount charged to the patient. Based on the summary statistics, the variable has 10000 data points, the average total charge is $5312.17. the standard deviation suggests that most values are within +- $2180 of the mean. The variable contains a wide range meaning a wide disparity between the smallest and largest charges.

```
# Also including details / statistics for the dependent variables of total charge
df['TotalCharge'].describe()
```

```
count    10000.000000
mean      5312.172769
std       2180.393838
min       1938.312067
25%       3179.374015
50%       5213.952000
75%       7459.699750
max       9180.728000
Name: TotalCharge, dtype: float64
```

Dependent Variables chosen for analysis:

- Gender - Data Type: Category, Categorical Variable – The patient's gender. Based on the summary statistics, the variable has 10000 data points containing 3 unique values with Female being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Gender"]].describe()
print(summary_stats)
```

```
        Gender
count    10000
unique       3
top     Female
freq      5018
```

- Initial_admin - Data Type: Category, Categorical Variable – Describes how the patient was admitted to the hospital initially. Based on summary statistics the variable has 10000 data points containing 3 unique values with Emergency Admission being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Initial_admin"]].describe()
print(summary_stats)
```

```
              Initial_admin
count                 10000
unique                    3
top     Emergency Admission
freq                   5060
```

- Complication_risk - Data Type: Category, Categorical Variable – Level of complication risk for the patient as assessed by a primary assessment. Based on summary statistics the variable has 10000 data points containing 3 unique values with Medium being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Complication_risk"]].describe()
print(summary_stats)
```

```
       Complication_risk
count              10000
unique                 3
top               Medium
freq                4517
```

- Services - Data Type: Category, Categorical Variable – Primary services the patient received. Based on summary statistics the variable has 10000 data points containing 4 unique values with Blood Work being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Services"]].describe()
print(summary_stats)
```

```
          Services
count        10000
unique           4
top     Blood Work
freq          5265
```

- HighBlood - Data Type: Category, Categorical Variable – Whether the patient has high blood pressure or not. Based on summary statistics the variable has 10000 data points containing 2 unique values with 0 being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["HighBlood"]].describe()
print(summary_stats)
```

```
        HighBlood
count       10000
unique          2
top             0
freq         5910
```

- Stroke - Data Type: Category, Categorical Variable – Whether the patient has had a stroke or not. Based on summary statistics the variable has 10000 data points containing 2 unique values with 0 being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Stroke"]].describe()
print(summary_stats)
```

```
        Stroke
count    10000
unique       2
top          0
freq      8007
```

- Overweight - Data Type: Category, Categorical Variable – Whether the patient is overweight or not. Based on summary statistics the variable has 10000 data points containing 2 unique values with 1 being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Overweight"]].describe()
print(summary_stats)
```

```
        Overweight
count        10000
unique           2
top              1
freq          7094
```

- Arthritis - Data Type: Category, Categorical Variable – Whether the patient has arthritis or not. Based on summary statistics the variable has 10000 data points containing 2 unique values with 0 being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Arthritis"]].describe()
print(summary_stats)
```

```
        Arthritis
count       10000
unique          2
top             0
freq         6426
```

- Diabetes - Data Type: Category, Categorical Variable – Whether the patient has diabetes or not. Based on summary statistics the variable has 10000 data points containing 2 unique values with 0 being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Diabetes"]].describe()
print(summary_stats)
```

```
        Diabetes
count      10000
unique         2
top            0
freq        7262
```

- Hyperlipidemia - Data Type: Category, Categorical Variable – Whether the patient has hyperlipidemia or not. Based on summary statistics the variable has 10000 data points containing 2 unique values with 0 being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Hyperlipidemia"]].describe()
print(summary_stats)
```

```
        Hyperlipidemia
count            10000
unique               2
top                  0
freq              6628
```

- BackPain - Data Type: Category, Categorical Variable – Whether the patient has back pain or not. Based on summary statistics the variable has 10000 data points containing 2 unique values with 0 being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["BackPain"]].describe()
print(summary_stats)
```

```
        BackPain
count      10000
unique         2
top            0
freq        5886
```

- Reflux_esophagitis - Data Type: Category, Categorical Variable – Whether the patient has reflux esophagitis or not. Based on summary statistics the variable has 10000 data points containing 2 unique values with 0 being the top one.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Reflux_esophagitis"]].describe()
print(summary_stats)
```

```
        Reflux_esophagitis
count                10000
unique                   2
top                      0
freq                  5865
```

- Age - Data Type: Int64, Continuous Variable – The patient's current age. Based on summary statistics the variable has 10000 data points containing a mean of 53, the standard deviation of 20 meaning that the standard deviation from the mean age is +- 20. And the max-age admitted is 89.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["Age"]].describe()
print(summary_stats)
```

```
                Age
count  10000.000000
mean      53.511700
std       20.638538
min       18.000000
25%       36.000000
50%       53.000000
75%       71.000000
max       89.000000
```

- VitD_levels Data Type: Float64, Continuous Variable – The patient's Vitamin D levels. Based on summary statistics the variable has 10000 data points containing a mean of 17 with a standard deviation from the mean being +- 2, and max vitamin D levels being 26.

```
# Including summary statistics for each of the variables chosen for analysis
summary_stats = df[["VitD_levels"]].describe()
print(summary_stats)
```

```
        VitD_levels
count  10000.000000
mean      17.964262
std        2.017231
min        9.806483
25%       16.626439
50%       17.951122
75%       19.347963
max       26.394449
```

C3, Univariate and Bivariate Analysis

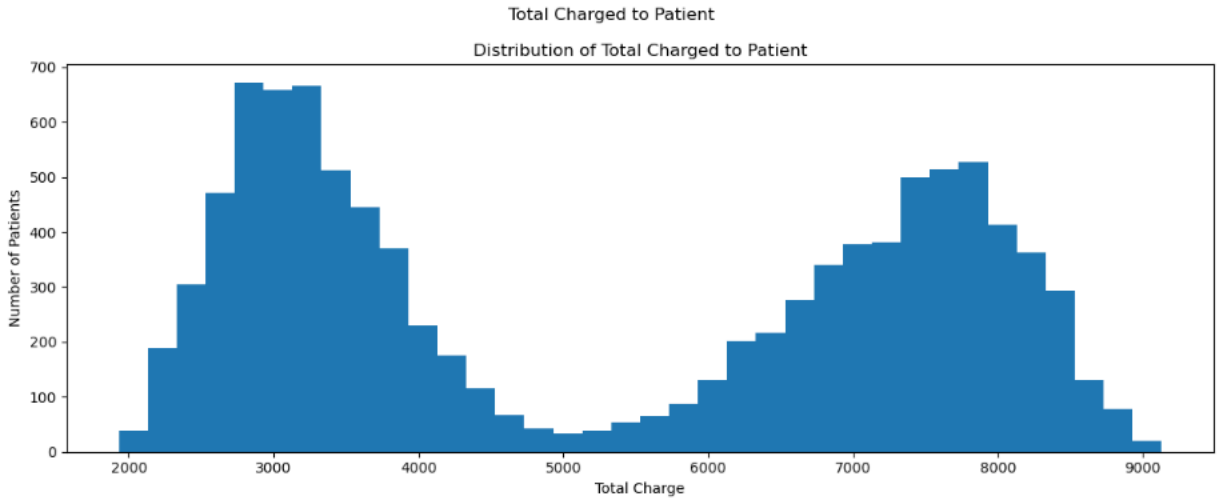
Univariate and bivariate statistics are techniques used to summarize and analyze a dataset that focuses on different aspects of the data. Univariate statistics involve analyzing a single variable at a time to describe the distribution and characterizing of that single variable without considering relationships with other variables. Univariate statistics help to understand central tendency, dispersion, shape, and outliers for a single variable. Whereas bivariate statistics involve the analysis of two variables to explore the relationship between them. Bivariate analysis can be used to determine whether and how two variables are related to each other.

Univariate statistics was performed on each of the chosen independent variables, bivariate statistics was performed on each of the independent variables against the dependent variable. A combination of charts was used to output these results. See the output of this analysis below.

The below chart shows the distribution of the Total Charged to patients.

```
# First examining the dependent variable for distribution of the Total Charged to patients.
# Min/Max values determined above used for the np.arrange function to arrange the data values.
plt.figure(figsize = [12,5])
plt.suptitle("Total Charged to Patient")
plt.title("Distribution of Total Charged to Patient")
bins = np.arange(1930, 9180, 200)
plt.hist(data=df, x="TotalCharge", bins=bins)
plt.xlabel("Total Charge")
plt.ylabel("Number of Patients")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



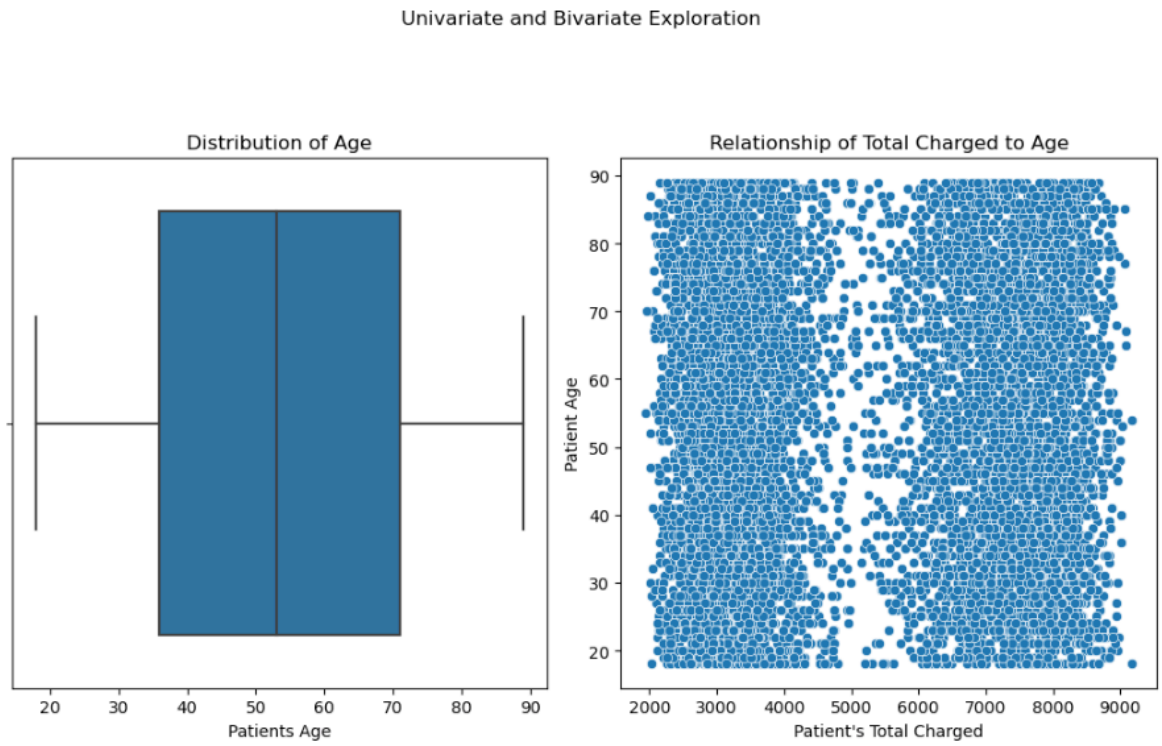
The below chart shows the distribution of the independent variable ‘Age’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Age’ and the dependent variable ‘TotalCharge’ using a Bivariate Analysis. The visualizations below show the median age being above 50 with most patients falling into the 35-70 age range. Additionally, patients age and total charge amount is widely scattered and does not indicate any direct relationship between the two variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Age  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
plt.suptitle("Univariate and Bivariate Exploration", y=1.05)

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot. This uses Age continuous variable in a bot pl
# Box Plot Analysis on Age.
sns.boxplot(x='Age', data=df, ax=axes[0])
axes[0].set_title('Distribution of Age')
axes[0].set_xlabel("Patients Age")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Age (continuous)
sns.scatterplot(x=df['TotalCharge'], y=model_df['Age'])
axes[1].set_title("Relationship of Total Charged to Age")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient Age")

# Set layout for charts.
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```



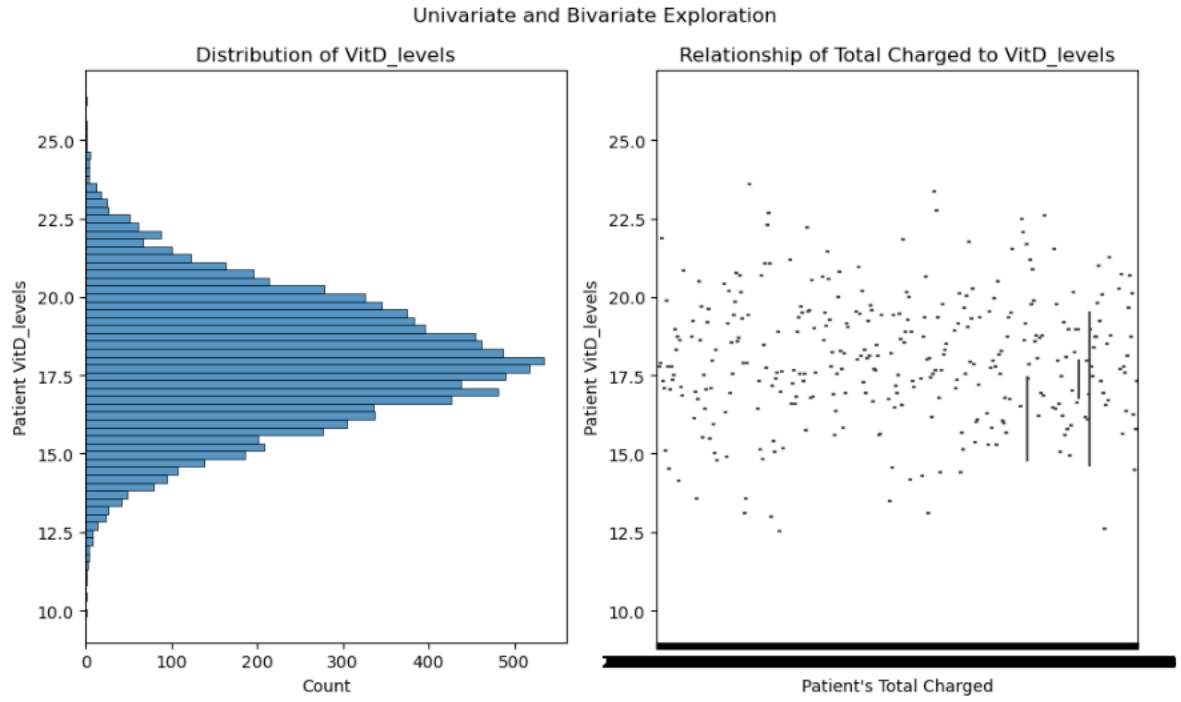
The below chart shows the distribution of the independent variable 'VitD_levels' using a Univariate analysis, as well as the distribution and correlation between the independent variable 'VitD_levels' and the dependent variable 'TotalCharge' using a Bivariate Analysis. The visualizations below show the even dispersion of the VitD_levels from the mean with most patients falling into the 17.5 level. Additionally, vitamin D levels and total charge are widely scattered and do not indicate any direct relationship between the two variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: VitD_levels  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Hist Plot Analysis on VitD_levels.
sns.histplot(data=df, y='VitD_levels',ax=axes[0])
axes[0].set_title('Distribution of VitD_levels')
axes[0].set_ylabel("Patient VitD_levels")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & VitD_levels (continuous)
sns.boxplot(data=df, x="TotalCharge", y="VitD_levels")
axes[1].set_title("Relationship of Total Charged to VitD_levels")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient VitD_levels")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



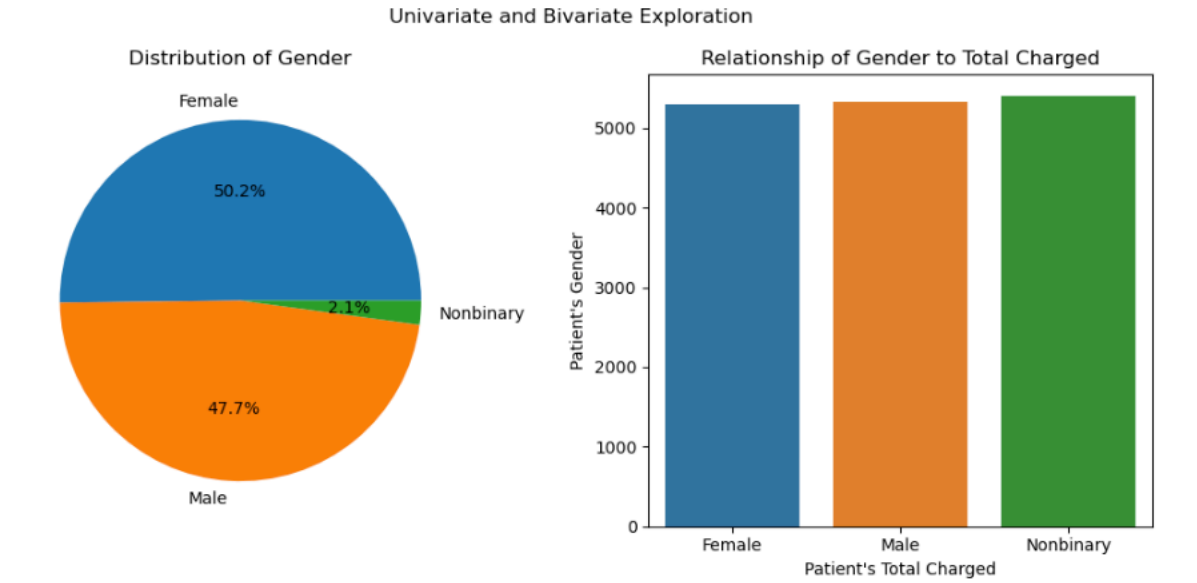
The below chart shows the distribution of the independent variable ‘Gender’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Gender’ and the dependent variable ‘TotalCharge’ using a Bivariate Analysis. The visualizations below show there is a similar number of Females and Males with very few non-binary patients. There is a tight dispersion between the categories indicating that Gender does not necessarily correlate with the total amount charged to the patient.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Gender  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Pie Chart Analysis on Gender.
df['Gender'].value_counts().plot.pie(autopct='%1.1f%%', ax=axes[0])
axes[0].set_title('Distribution of Gender')
axes[0].set_ylabel("")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Gender (categorical)
grouped_data = df.groupby('Gender')['TotalCharge'].mean().reset_index()
sns.barplot(x="Gender", y="TotalCharge", data=grouped_data)
axes[1].set_title("Relationship of Gender to Total Charged")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient's Gender")


# Set layout for charts.
plt.tight_layout()
plt.show()
```



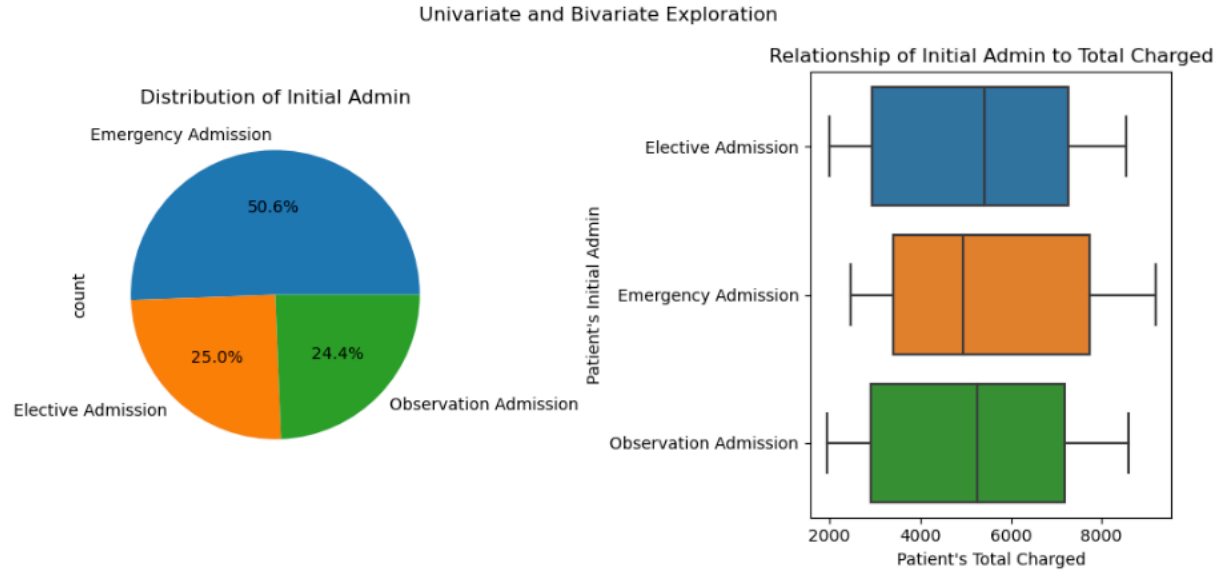
The below chart shows the distribution of the independent variable 'Initial_admin' using a Univariate analysis, as well as the distribution and correlation between the independent variable 'Initial_admin' and the dependent variable 'TotalCharge' using a Bivariate Analysis. The visualizations below show that most patients are admitted for emergency reasons, and patients who were admitted for an emergency tend to have a higher price dispersion than those who were not.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Initial_admin  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Pie Chart Analysis on Initial Admin.
df['Initial_admin'].value_counts().plot.pie(autopct='%1.1f%%', ax=axes[0])
axes[0].set_title('Distribution of Initial Admin')
axes[0].set_xlabel("")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Initial Admin (categorical)
sns.boxplot(data=df, x="TotalCharge", y="Initial_admin")
axes[1].set_title("Relationship of Initial Admin to Total Charged")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient's Initial Admin")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



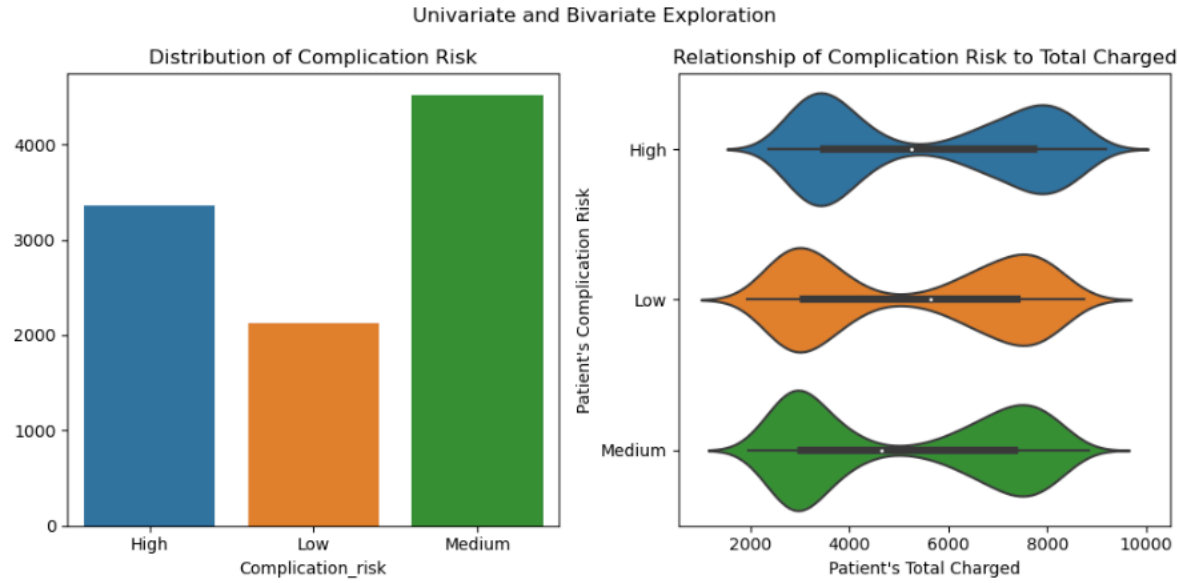
The below chart shows the distribution of the independent variable 'Complication_risk' using a Univariate analysis, as well as the distribution and correlation between the independent variable 'Complication_risk' and the dependent variable 'TotalCharge' using a Bivariate Analysis. The visualizations below show that most patients fall into the medium-risk category however there is a similar dispersion between the categories and patients with a high Complication risk tend to have a slightly higher total charge amount.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Complication_risk  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Count Plot Analysis on Complication risk.
sns.countplot(data=df, x='Complication_risk', ax=axes[0])
axes[0].set_title('Distribution of Complication Risk')
axes[0].set_ylabel("")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Complication risk (categorical)
sns.violinplot(data=df, x="TotalCharge", y="Complication_risk")
axes[1].set_title("Relationship of Complication Risk to Total Charged")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient's Complication Risk")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



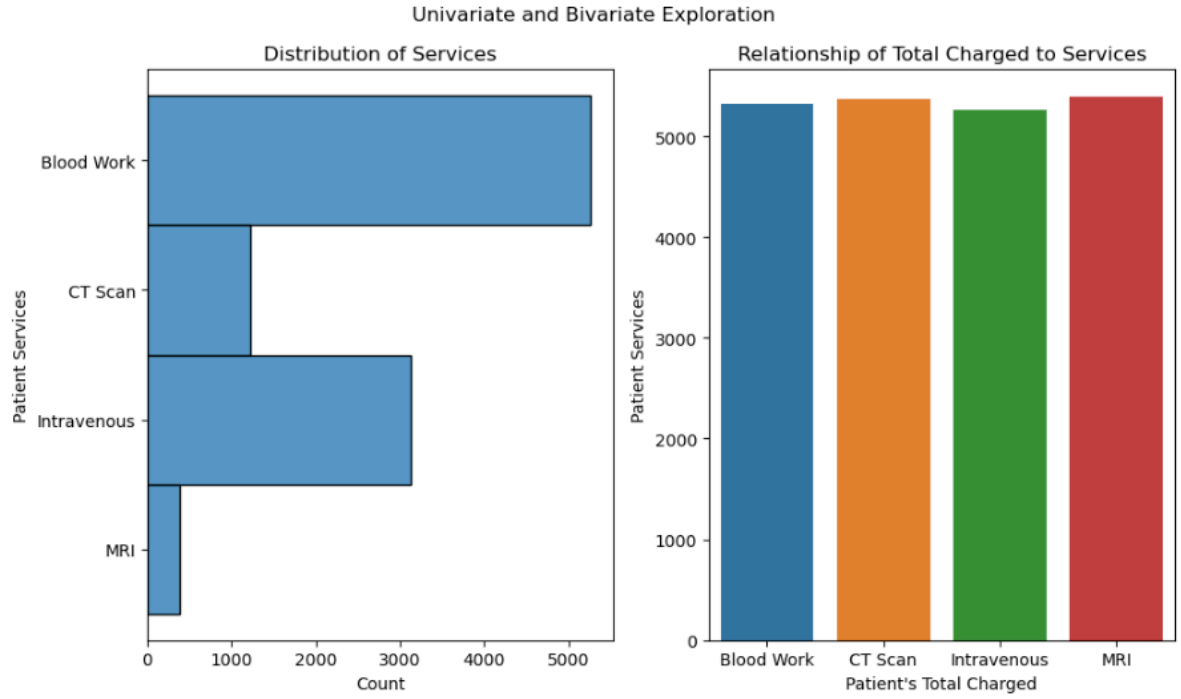
The below chart shows the distribution of the independent variable ‘Services’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Services’ and the dependent variable ‘TotalCharge’ using a Bivariate Analysis. The visualizations below show that most patients fall into the blood work and intravenous categories, however, there is a similar dispersion between the categories indicating that there is no single Service that has a huge influence on the total amount charged to the patient.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Services  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Count Plot Analysis on Services.
sns.histplot(data=df, y='Services',ax=axes[0])
axes[0].set_title('Distribution of Services')
axes[0].set_ylabel("Patient Services")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Services (categorical)
grouped_data = df.groupby('Services')['TotalCharge'].mean().reset_index()
sns.barplot(x='Services', y='TotalCharge', data=grouped_data)
axes[1].set_title("Relationship of Total Charged to Services")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient Services")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



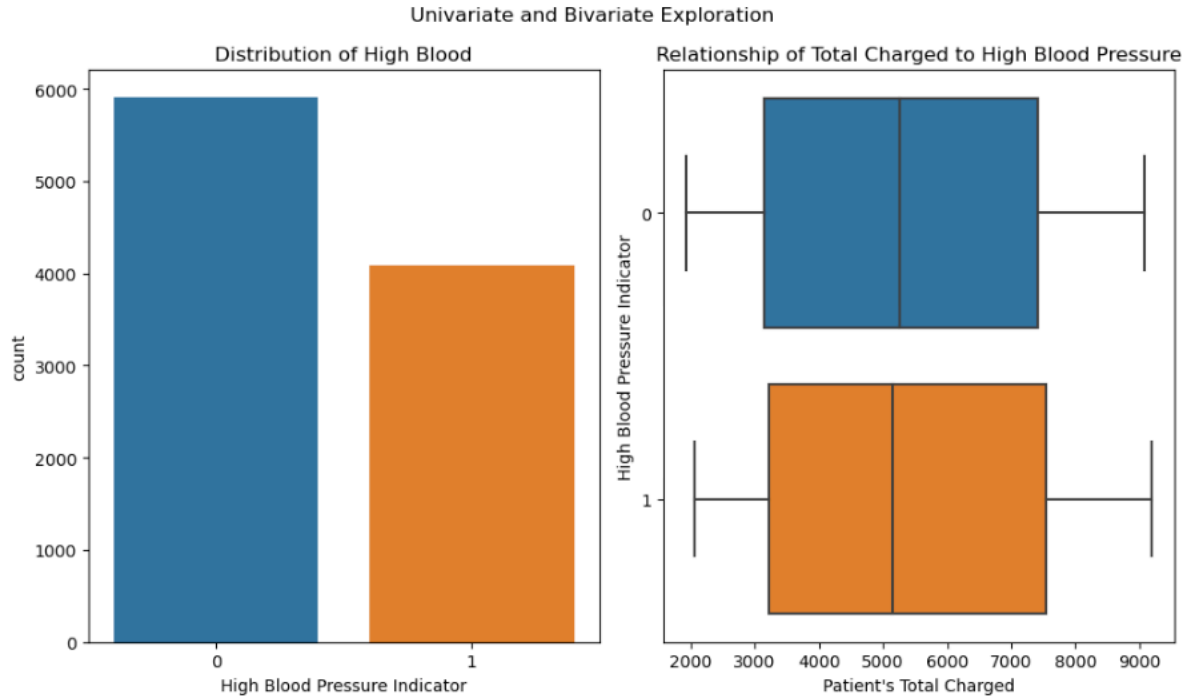
The below chart shows the distribution of the independent variable 'HighBlood' using a Univariate analysis, as well as the distribution and correlation between the independent variable 'HighBlood' and the dependent variable 'TotalCharge' using a Bivariate Analysis. The visualizations below show that there are more patients who do not have high blood pressure and there is a similar dispersion between the categories indicating that having high blood pressure vs not does not have a huge influence on the total amount charged to the patient.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: HighBlood  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Count Plot Analysis on HighBlood.
sns.countplot(data=df, x='HighBlood', ax=axes[0])
axes[0].set_title('Distribution of High Blood')
axes[0].set_xlabel("Count")
axes[0].set_xlabel("High Blood Pressure Indicator")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & HighBlood (categorical)
#grouped_data = df.groupby('HighBlood')['TotalCharge'].mean().reset_index()
sns.boxplot(data=df, x='TotalCharge', y='HighBlood')
axes[1].set_title("Relationship of Total Charged to High Blood Pressure")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("High Blood Pressure Indicator")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



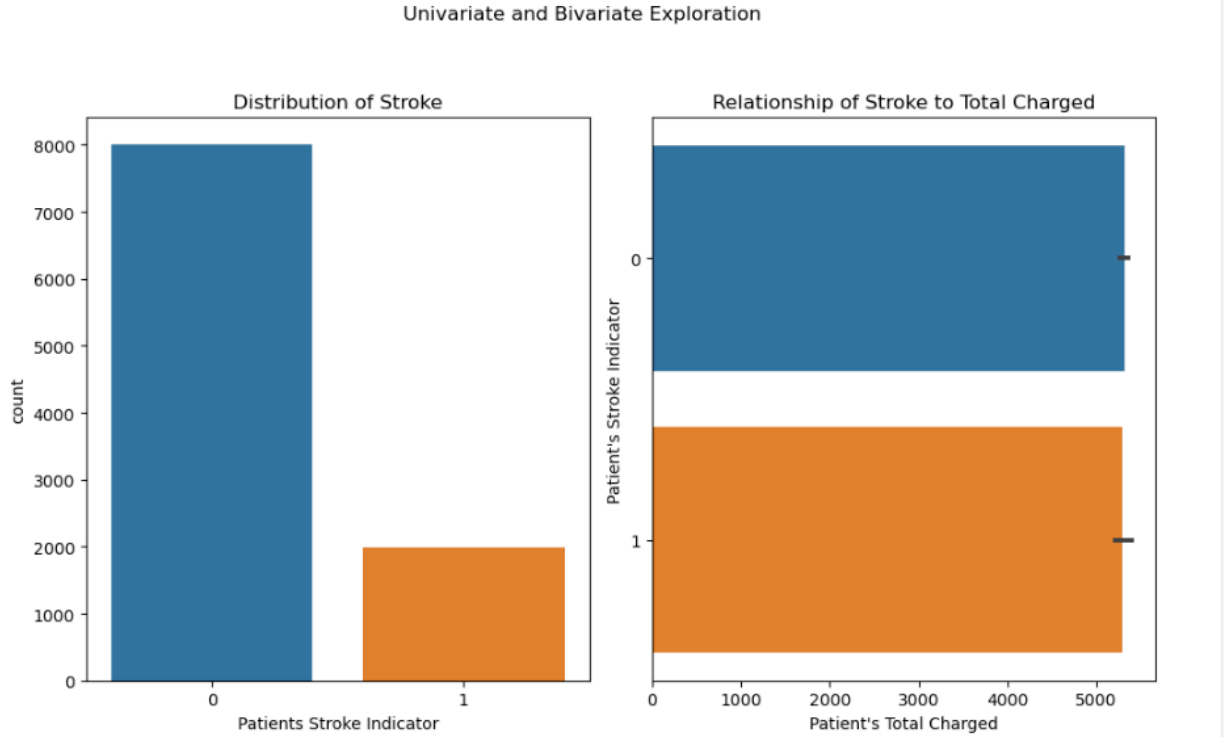
The below chart shows the distribution of the independent variable ‘Stroke’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Stroke’ and the dependent variable ‘TotalCharge’ using a Bivariate Analysis. The visualizations below show that more patients have not had a stroke and there is a similar dispersion between the categories indicating that patients who have had a stroke do not have a huge influence on the total amount charged to the patient.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Stroke  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
plt.suptitle("Univariate and Bivariate Exploration", y=1.05)

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot. This uses Age continuous variable in a bot pl
# Box Plot Analysis on Age.
sns.countplot(x='Stroke', data=df, ax=axes[0])
axes[0].set_title('Distribution of Stroke')
axes[0].set_xlabel("Patients Stroke Indicator")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Complication risk (categorical)
sns.barplot(data=df, x="TotalCharge", y="Stroke")
axes[1].set_title("Relationship of Stroke to Total Charged")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient's Stroke Indicator")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



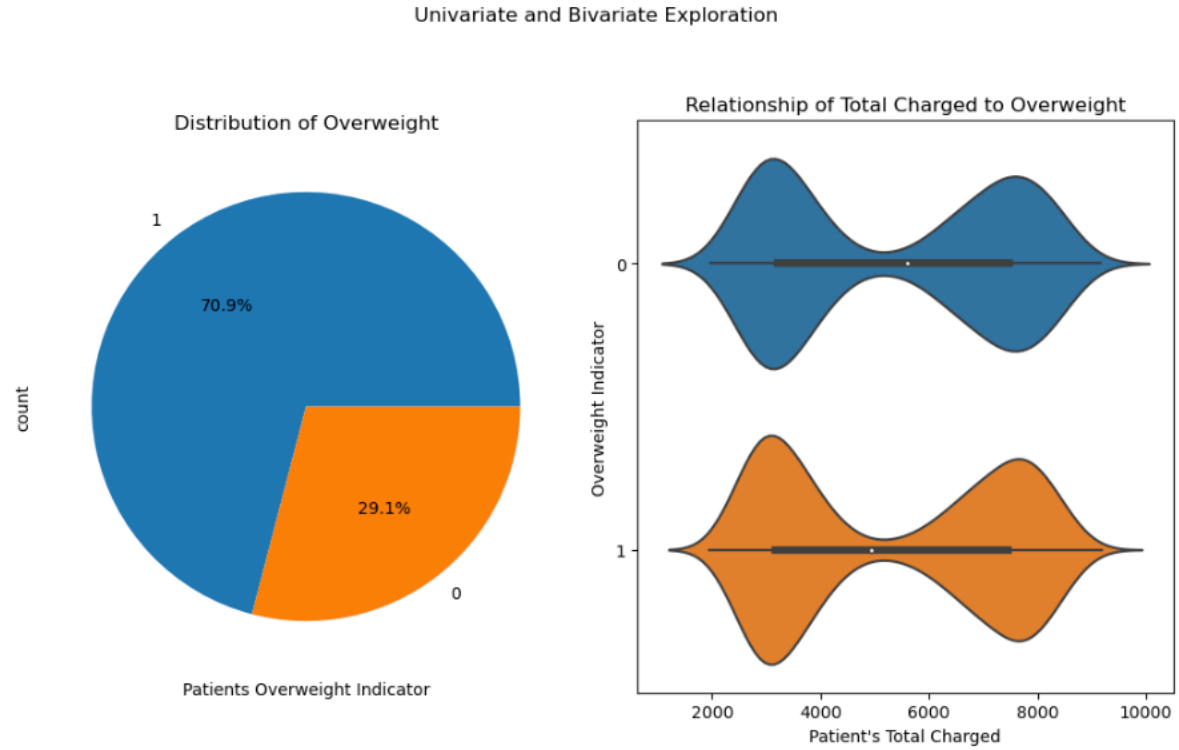
The below chart shows the distribution of the independent variable ‘Overweight’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Overweight’ and the dependent variable ‘TotalCharge’ using a Bivariate Analysis. The visualizations below show that more patients are overweight and there is a similar dispersion between the categories indicating that patients who are or are not overweight do not have a huge influence on the total amount charged to the patient.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Overweight  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 6))
plt.suptitle("Univariate and Bivariate Exploration", y=1.05)

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot. This uses Age continuous variable in a bot pl
# Box Plot Analysis on Overweight.
df['Overweight'].value_counts().plot.pie(autopct='%1.1f%%', ax=axes[0])
axes[0].set_title('Distribution of Overweight')
axes[0].set_xlabel("Patients Overweight Indicator")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & HighBlood (categorical)
#grouped_data = df.groupby('HighBlood')['TotalCharge'].mean().reset_index()
sns.violinplot(data=df, x='TotalCharge', y='Overweight')
axes[1].set_title("Relationship of Total Charged to Overweight")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Overweight Indicator")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



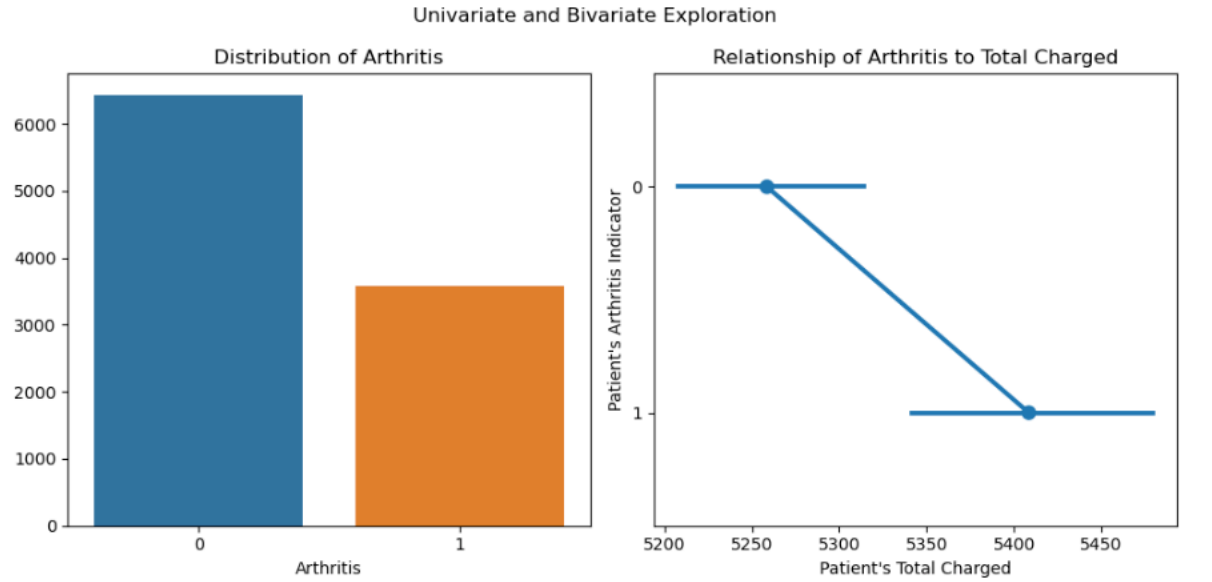
The below chart shows the distribution of the independent variable ‘Arthritis’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Arthritis’ and the dependent variable ‘TotalCharge’ using a Bivariate Analysis. The visualizations below show that there are more patients who do not have arthritis and that patients who have arthritis overall tend to have a higher bill amount then those who do not.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Arthritis  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Count Plot Analysis on Arthritis.
sns.countplot(data=df, x='Arthritis', ax=axes[0])
axes[0].set_title('Distribution of Arthritis')
axes[0].set_ylabel("")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Arthritis (categorical)
sns.pointplot(data=df, x="TotalCharge", y="Arthritis")
axes[1].set_title("Relationship of Arthritis to Total Charged")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient's Arthritis Indicator")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



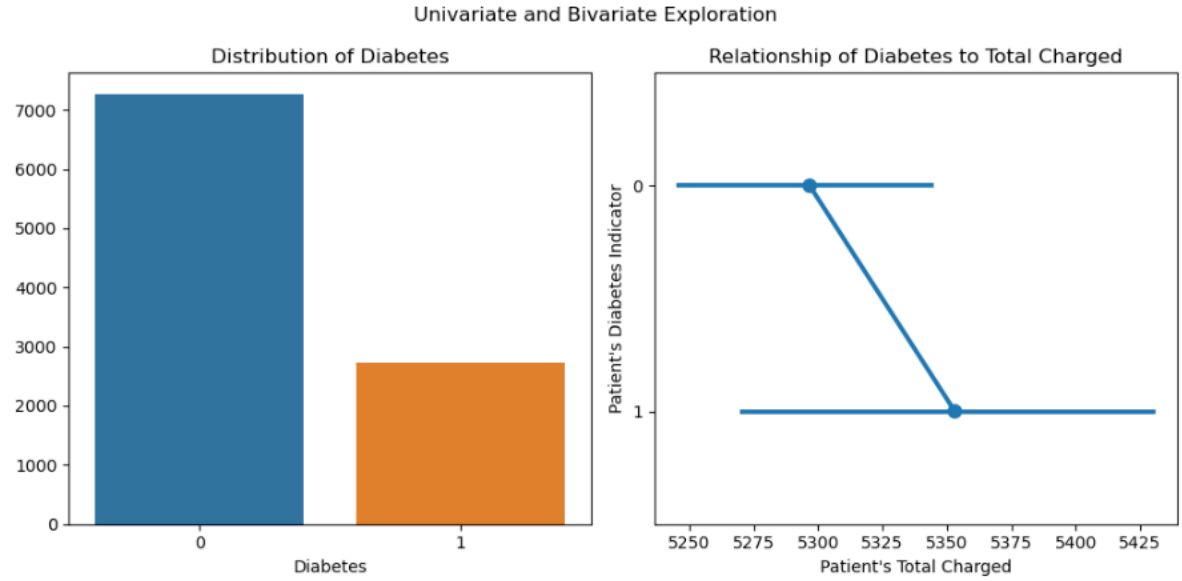
The below chart shows the distribution of the independent variable ‘Stroke’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Stroke’ and the dependent variable ‘TotalCharge’ using a Bivariate Analysis. The visualizations below show that more patients have not had a stroke and there is a similar dispersion between the categories indicating that having high blood pressure vs not does not have a huge influence on the total amount charged to the patient.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Diabetes  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Count Plot Analysis on Arthritis.
sns.countplot(data=df, x='Diabetes', ax=axes[0])
axes[0].set_title('Distribution of Diabetes')
axes[0].set_ylabel("")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Arthritis (categorical)
sns.pointplot(data=df, x="TotalCharge", y="Diabetes")
axes[1].set_title("Relationship of Diabetes to Total Charged")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient's Diabetes Indicator")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



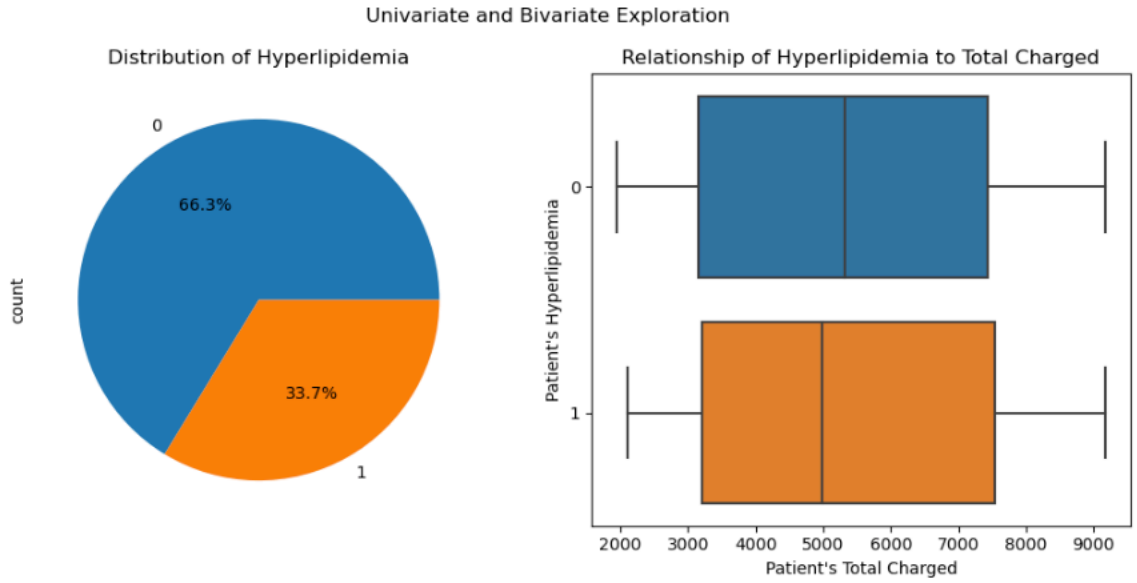
The below chart shows the distribution of the independent variable ‘Hyperlipidemia’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Hyperlipidemia’ and the dependent variable ‘TotalCharge’ using a Bivariate Analysis. The visualizations below show that there are more patients without hyperlipidemia and there is a similar dispersion between the categories indicating that hyperlipidemia does not have a huge influence on the total amount charged to the patient.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Hyperlipidemia  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Pie Chart Analysis on Hyperlipidemia.
df['Hyperlipidemia'].value_counts().plot.pie(autopct='%1.1f%%', ax=axes[0])
axes[0].set_title('Distribution of Hyperlipidemia')
axes[0].set_xlabel("")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Hyperlipidemia (categorical)
sns.boxplot(data=df, x="TotalCharge", y="Hyperlipidemia")
axes[1].set_title("Relationship of Hyperlipidemia to Total Charged")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient's Hyperlipidemia")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



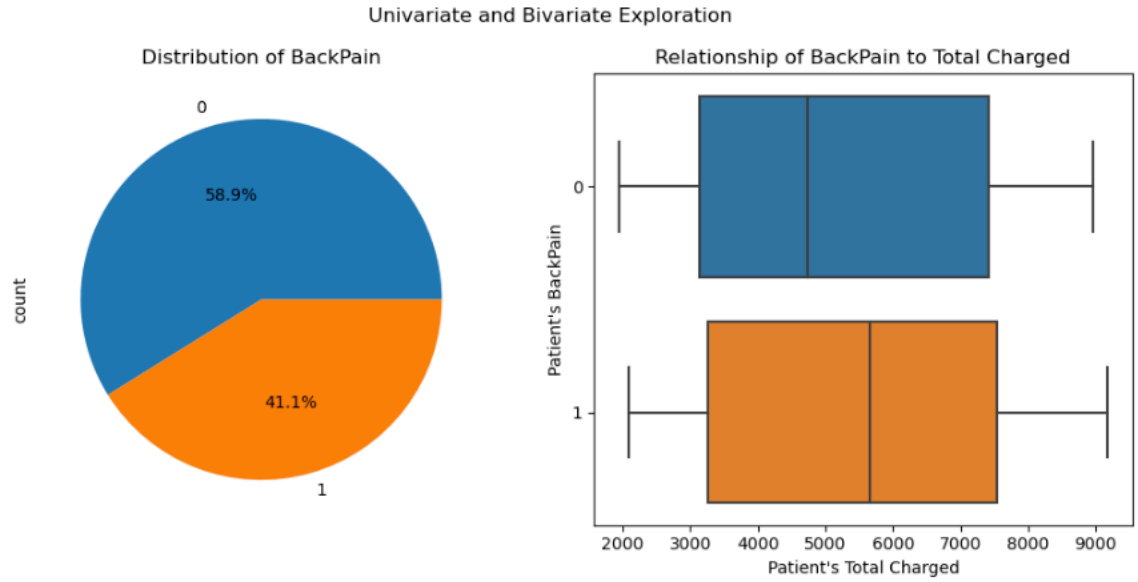
The below chart shows the distribution of the independent variable ‘BackPain’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘BackPain’ and the dependent variable ‘TotalCharge’ using a Bivariate Analysis. The visualizations below show that there are more patients without back pain, however, having back pain has a slightly higher overall price dispersion than those without back pain.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: BackPain  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Pie Chart Analysis on Hyperlipidemia.
df['BackPain'].value_counts().plot.pie(autopct='%1.1f%%', ax=axes[0])
axes[0].set_title('Distribution of BackPain')
axes[0].set_xlabel("")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & BackPain (categorical)
sns.boxplot(data=df, x="TotalCharge", y="BackPain")
axes[1].set_title("Relationship of BackPain to Total Charged")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient's BackPain")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



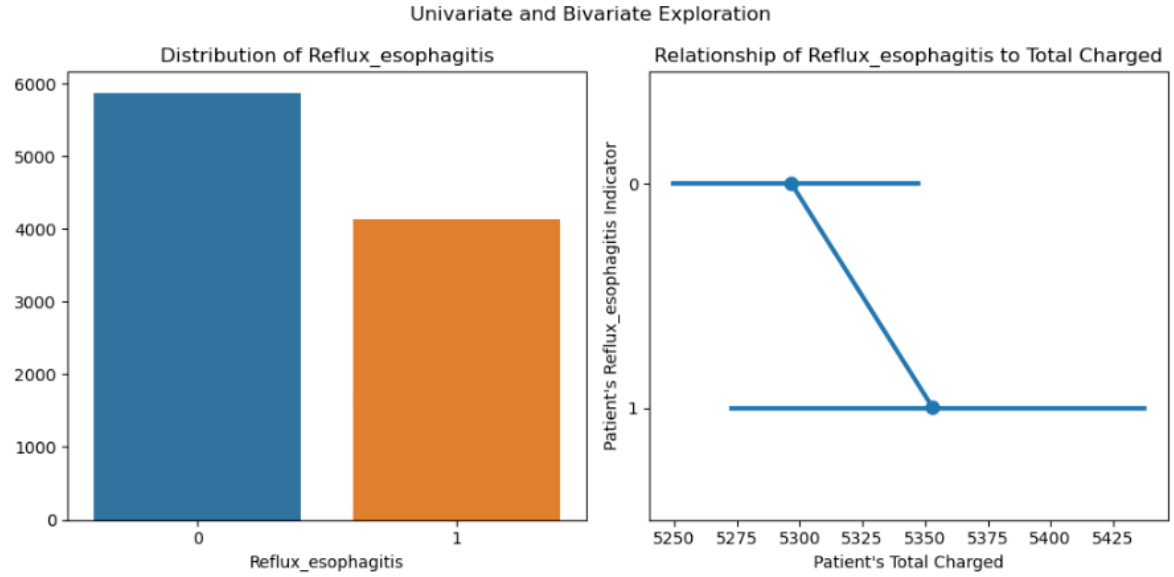
The below chart shows the distribution of the independent variable 'Reflux_esophagitis' using a Univariate analysis, as well as the distribution and correlation between the independent variable 'Reflux_esophagitis' and the dependent variable 'TotalCharge' using a Bivariate Analysis. The visualizations below show that there are more patients without Reflux esophagitis, however, patients who have Reflux esophagitis tend to have a higher overall price than those who do not.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Reflux_esophagitis  Dependent Variable: TotalCharge
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
plt.suptitle("Univariate and Bivariate Exploration")

# LEFT plot: this divides the figure into a 1x2 grid and selects the second subplot.
# Count Plot Analysis on Reflux_esophagitis.
sns.countplot(data=df, x='Reflux_esophagitis', ax=axes[0])
axes[0].set_title('Distribution of Reflux_esophagitis')
axes[0].set_ylabel("")

# RIGHT plot: Bivariate exploration of Total Charge (continuous) & Reflux_esophagitis (categorical)
sns.pointplot(data=df, x="TotalCharge", y="Diabetes")
axes[1].set_title("Relationship of Reflux_esophagitis to Total Charged")
axes[1].set_xlabel("Patient's Total Charged")
axes[1].set_ylabel("Patient's Reflux_esophagitis Indicator")

# Set layout for charts.
plt.tight_layout()
plt.show()
```



C4, Data Transformation

The first step in the data cleaning and transformation steps, as initially described in C1, is to identify where all the inconsistencies are, the dataset was reviewed for duplicates, nulls, outliers, data types, and unique values. .info() was called to find if there were any null values and to get an overview of the dataset, no null values were identified. To find duplicate values, the first step is to assess the number of unique values the data set should contain. As described in the data dictionary there are 10000 records, CustomerID is a unique value that should contain no duplicates. .duplicated() was used to print the duplicates in the dataset, and no duplicates were found. To identify if there are any missing values, isnull().sum() was used to determine this, there were no missing values found. Next, .describe() was used to determine if there were any outliers in the data, no outliers were detected. I could recall that the time zone has outliers in D207, however this appears to have been cleaned and consists of unique time zones. Dtypes were called to identify all the incorrect data types in the dataset. As part of this analysis, all categorical variables must be converted from the data type of object to category. Lastly, value_counts() was called to

identify where unique values were, it appears there are 10000 unique values inside the dataset. See the code below showing the outputs of the functions used to observe the dataset.

```
: # Using pandas to read the loaded csv file, indicating the first column is an index which pandas does not need.
  df = pd.read_csv('./medical_clean-Copy1.csv', index_col=0)
  # Checking the info on the loaded dataset, indicating columns, data types, and size.
  df.info()
```

```
<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
```

```
# Exploring the first few rows of data using .head() function to get a feel for the data
df.head()
```

CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Population	...
1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL	Morgan	35621	34.34960	-86.72508	2951	...
2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL	Jackson	32446	30.84513	-85.22907	11303	...
3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD	Minnehaha	57110	43.54321	-96.63772	17125	...
4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN	Waseca	56072	43.89744	-93.51479	2162	...
5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA	King William	23181	37.59894	-76.88958	5287	...

5 rows × 49 columns

```
# Checking Dataframe for duplication issues -- boolean output indicating if rows are duplicated or now.
duplicates = df[df.duplicated()]
print(duplicates)
```

```
Empty DataFrame
Columns: [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, Compli
cation_risk, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagitis, Asthma, Serv
ices, Initial_days, TotalCharge, Additional_charges, Item1, Item2, Item3, Item4, Item5, Item6, Item7, Item8]
Index: []

[0 rows x 49 columns]
```

```
# Identifying any duplicate values in the unique field Customer ID - course notes indicate there are 10,000 customers
# If no duplicates in the Customer_ID field then any duplicates above are expected and ok to have.
df.duplicated(subset=['Customer_id']).sum()
```

```
0
```

```
# Identify any missing values in the dataset -- no missing values identified.
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
```

```
# Identify outliers in fields
df.describe()
```

TotalCharge	Additional_charges	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8
000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
5312.172769	12934.528587	3.518800	3.506700	3.511100	3.515100	3.496900	3.522500	3.494000	3.509700
2180.393838	6542.601544	1.031966	1.034825	1.032755	1.036282	1.030192	1.032376	1.021405	1.042312
1938.312067	3125.703000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
3179.374015	7986.487755	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000
5213.952000	11573.977735	4.000000	3.000000	4.000000	4.000000	3.000000	4.000000	3.000000	3.000000
7459.699750	15626.490000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000
9180.728000	30566.070000	8.000000	7.000000	8.000000	7.000000	7.000000	7.000000	7.000000	7.000000

```
# Identify all incorrect data types in the dataset
df.dtypes
```

```
Customer_id            object
Interaction            object
UID                    object
City                   object
State                  object
County                 object
Zip                     int64
Lat                   float64
Lng                   float64
Population              int64
Area                   object
TimeZone               object
Job                    object
Children                int64
Age                     int64
Income                float64
Marital                object
Gender                 object
ReAdmis                object
VitD_levels           float64
Doc_visits              int64
Full_meals_eaten        int64
vitD_supp               int64
Soft_drink             object
Initial_admin          object
HighBlood              object
Stroke                 object
Complication_risk      object
Overweight             object
Arthritis              object
Diabetes               object
Hyperlipidemia         object
BackPain               object
Anxiety                object
Allergic_rhinitis      object
Reflux_esophagitis     object
Asthma                 object
Services               object
Initial_days          float64
TotalCharge           float64
Additional_charges    float64
Item1                   int64
Item2                   int64
Item3                   int64
Item4                   int64
Item5                   int64
Item6                   int64
Item7                   int64
Item8                   int64
dtype: object
```

```
# Identify where unique values are
df.value_counts()
```

```
Customer_id  Interaction                            UID                              City             State  County        Zip    L
at       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  Complic
ation_risk  Overweight  Arthritis  Diabetes  Hyperlipidemia  BackPain  Anxiety  Allergic_rhinitis  Reflux_esophagitis  Asthma  Servi
ces     Initial_days  TotalCharge  Additional_charges  Item1  Item2  Item3  Item4  Item5  Item6  Item7  Item8
A01882       da8650df-5ada-4a78-b276-88ec0f2707c1  11361eef68a2da28a0f0b9e0792a54b1  Wahpeton         ND     Richland      58076  4
6.27285  -96.60753   355         Suburban  America/Chicago      Professor Emeritus             1         30   80568.37  Never Marr
ied  Male    No       16.829192    6           1                 0          No          Emergency Admission    Yes         No      Hi
gh              Yes         No         No        Yes             No        Yes      Yes               No                 No
Intravenous  4.921720      3537.787245  10911.506110        5      4      4      4      3      5      5      5        1
R262790      c46195fe-0058-4168-b327-34dc4cf3befd  919eb82ff47c7fecfc61d92ba0c58d1e  Kansas City      MO     Clay          64116  3
9.14879  -94.57455   15576       Urban     America/Chicago      Biomedical engineer            2         56   55958.21  Widowed
Male    No       17.664165    5           3                 0          No          Elective Admission     No          No      High
Yes         Yes        No         No          No        Yes         No                No                 No      Blood Work   18.7
70204     3965.235362  9838.295836         4      4      2      4      4      2      4      4        1
R243544      631ec2bc-e058-4956-bcd9-87b82542f96c  46c733d95d8022a160c418c84d89c9ea  Tama             IA     Tama          52339  4
1.92296  -92.58861   4213        Rural     America/Chicago      Radio producer                 0         73   7449.97   Widowed
Male    Yes      13.436650    6           1                 0          Yes         Elective Admission     No          Yes     Low
No          No         No         No          No        Yes      No                No                 Yes     Blood Work   67.7
96180     7496.873000  12058.210000        4      3      4      2      4      5      3      3        1
R247747      71e9631c-8393-4cd9-aa42-bcb9d71603a1  dfc3bee4e8fd2ce863dc34677bd8136c  Orlando          FL     Orange        32824  2
8.38842  -81.34887   46545       Rural     America/New_York     Learning mentor                1         41   50794.63  Divorced
Female  Yes      17.965560    6           1                 0          No          Elective Admission     No          No      High
Yes         Yes        Yes        No          Yes       Yes      No                No                 No      Blood Work   68.4
73020     8198.126000  7388.460000         2      2      2      5      4      3      3      3        1
R249440      8156b235-cda9-4a54-abf9-fd82eedecd21  528f238026c50d476936fec04933818c  Westover         PA     Clearfield    16692  4
0.75602  -78.72004   674         Urban     America/New_York     Producer, television/film/video  0         72   32559.51  Widowed
Male    No       16.182905    4           3                 0          No          Observation Admission  No          No      High
No          No         No         No          No        No       Yes               No                 Yes     Blood Work   9.31
9670      3093.394945  12346.024090        3      3      3      5      2      5      4      5        1

..
I580267      f2f76538-090b-474a-a5de-772d4fef9b99  14af97d24736aabab451418a897e8edb  High Bridge      NJ     Hunterdon     8829   4
0.66802  -74.89420   3558        Urban     America/New_York     Leisure centre manager         0         68   49215.82  Widowed
Male    No       14.981662    5           1                 0          Yes         Observation Admission  No          No      Medium
Yes         No         Yes       No          Yes       No       Yes               No                 No      Blood Work   2.75
4235      2302.292170  11365.587090        3      3      4      5      2      4      4      3        1
I580789      74116271-ffc6-4308-9794-51104a5fe117  f0f32de8d7802a0cc8b999ed3841d15a  Huntington Beach  CA     Orange        92647  3
3.72350  -118.00674  62718       Suburban  America/Los_Angeles  Contractor                     3         35   73928.85  Never Marr
ied  Male    No       17.309715    5           1                 0          No          Elective Admission     No          No      Lo
w               Yes         Yes        No          Yes             Yes        No        Yes               No
Intravenous  10.450094     3083.292179  5954.492709         3      3      2      4      4      2      3      4        1
I587810      3de73d8a-e063-435e-84b2-4a43ade4377e  2a9926db79ef6e549d8645e3d45c462c  Ringgold         VA     Pittsylvania  24586  3
6.60700  -79.28300   6131        Urban     America/New_York     Manufacturing engineer         2         62   5667.36   Never Marr
ied  Female  No       15.310240    3           2                 0          No          Emergency Admission    No          No      Me
dium            No          Yes        No        No              Yes       Yes      No                Yes                No
Blood Work   43.505940     6235.683000  10722.930000        4      3      3      3      4      3      4      3        1
I590981      b3ca6ce6-33a6-46d9-ad0e-38ce27555ce9  2478d49f0eac00558560f13bd048b00a  Bluejacket       OK     Craig         74333  3
6.79148  -95.08617   1129        Urban     America/Chicago      Neurosurgeon                   1         32   2905.77   Married
Male    No       23.941426    5           0                 0          No          Emergency Admission    No          No      High
Yes         No         No         No          No        No       Yes               Yes                Yes     Blood Work   2.84
0540      3134.512364  6367.059063         5      5      5      4      2      6      4      6        1
Z996563      c630c6a0-e720-4b3c-b90d-74fabbbd1de9  75c5c551c82986f6e3188b6e7a396ebe  Lansing          KS     Leavenworth   66043  3
9.25162  -94.87840   11117       Urban     America/Chicago      Surveyor, mining               0         65   13179.07  Separated
Male    No       19.579960    5           1                 0          No          Elective Admission     Yes         Yes     High
No          No         No         No          No        No       No                Yes                No      Blood Work   49.2
71380     6478.374000  21770.360000        3      3      2      4      4      3      3      3        1
Name: count, Length: 10000, dtype: int64
```

```
# Checking Timezones to confirm there are 26 unique ones.
df['TimeZone'].describe()
```

```
count                10000
unique                  26
top       America/New_York
freq                  3889
Name: TimeZone, dtype: object
```

After identifying where the inconsistencies are, the next step would be to proceed with cleaning the data to ensure the data is adequate and ready for analysis. The data appeared overall 'clean', however, there are a few items we can change to make the data ready for analysis. First, we need to convert the necessary data types to categories. All Boolean 'Yes' and 'No' variables should

be converted to category, and all categorical variables need to be converted from object to category. Additionally, it was observed that there were a few other datatypes that were identified as incorrect which we can clean up. Lastly, a few fields have 6 decimal points which would be cleaner only having 2, these were re-expressed to reflect this change. See this performed below.

```
# Convert all boolean Yes or No to numerical categories
bool_map = {"Yes" : 1, "No" : 0}
df['ReAdmis'] = df['ReAdmis'].map(bool_map)
df['Soft_drink'] = df['Soft_drink'].map(bool_map)
df['HighBlood'] = df['HighBlood'].map(bool_map)
df['Stroke'] = df['Stroke'].map(bool_map)
df['Overweight'] = df['Overweight'].map(bool_map)
df['Arthritis'] = df['Arthritis'].map(bool_map)
df['Diabetes'] = df['Diabetes'].map(bool_map)
df['Hyperlipidemia'] = df['Hyperlipidemia'].map(bool_map)
df['BackPain'] = df['BackPain'].map(bool_map)
df['Anxiety'] = df['Anxiety'].map(bool_map)
df['Allergic_rhinitis'] = df['Allergic_rhinitis'].map(bool_map)
df['Reflux_esophagitis'] = df['Reflux_esophagitis'].map(bool_map)
df['Asthma'] = df['Asthma'].map(bool_map)
```

```
# Convert object variables - object to category
df['City'] = df['City'].astype('category')
df['State'] = df['State'].astype('category')
df['County'] = df['County'].astype('category')
df['Area'] = df['Area'].astype('category')
df['TimeZone'] = df['TimeZone'].astype('category')
df['Job'] = df['Job'].astype('category')
df['Marital'] = df['Marital'].astype('category')
df['Gender'] = df['Gender'].astype('category')
df['Initial_admin'] = df['Initial_admin'].astype('category')
df['Complication_risk'] = df['Complication_risk'].astype('category')
df['Services'] = df['Services'].astype('category')
df['ReAdmis'] = df['ReAdmis'].astype('category')
df['Soft_drink'] = df['Soft_drink'].astype('category')
df['HighBlood'] = df['HighBlood'].astype('category')
df['Stroke'] = df['Stroke'].astype('category')
df['Overweight'] = df['Overweight'].astype('category')
df['Arthritis'] = df['Arthritis'].astype('category')
df['Diabetes'] = df['Diabetes'].astype('category')
df['Hyperlipidemia'] = df['Hyperlipidemia'].astype('category')
df['BackPain'] = df['BackPain'].astype('category')
df['Anxiety'] = df['Anxiety'].astype('category')
df['Allergic_rhinitis'] = df['Allergic_rhinitis'].astype('category')
df['Reflux_esophagitis'] = df['Reflux_esophagitis'].astype('category')
df['Asthma'] = df['Asthma'].astype('category')
df['Item1'] = df['Item1'].astype('category')
df['Item2'] = df['Item2'].astype('category')
df['Item3'] = df['Item3'].astype('category')
df['Item4'] = df['Item4'].astype('category')
df['Item5'] = df['Item5'].astype('category')
df['Item6'] = df['Item6'].astype('category')
df['Item7'] = df['Item7'].astype('category')
df['Item8'] = df['Item8'].astype('category')

# Change all datatypes identified as incorrect
df['Zip'] = df['Zip'].astype(object)
df['Children'] = df['Children'].astype(int)
df['Age'] = df['Age'].astype(int)
```

```
# Noticed the charge fields have 6 decimal places, it would be cleaner to have 2. Updating th
df['TotalCharge'].round(2)
df['Additional_charges'].round(2)
df['Initial_days'].round(2)
df['Income'].round(2)
```

After initial inconsistencies have been cleaned up we can provide evidence of the now cleaned dataset, see below.

```
: # Providing evidence of the now clean dataset
  df.info()
```

```
<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  category
 4   State               10000 non-null  category
 5   County              10000 non-null  category
 6   Zip                 10000 non-null  object
 7   Lat                 10000 non-null  float64
 8   Lng                 10000 non-null  float64
 9   Population          10000 non-null  int64
 10  Area                10000 non-null  category
 11  TimeZone            10000 non-null  category
 12  Job                 10000 non-null  category
 13  Children            10000 non-null  int64
 14  Age                 10000 non-null  int64
 15  Income              10000 non-null  float64
 16  Marital             10000 non-null  category
 17  Gender              10000 non-null  category
 18  ReAdmis             10000 non-null  category
 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  category
 24  Initial_admin       10000 non-null  category
 25  HighBlood           10000 non-null  category
 26  Stroke              10000 non-null  category
 27  Complication_risk   10000 non-null  category
 28  Overweight          10000 non-null  category
 29  Arthritis           10000 non-null  category
 30  Diabetes            10000 non-null  category
 31  Hyperlipidemia      10000 non-null  category
 32  BackPain            10000 non-null  category
 33  Anxiety             10000 non-null  category
 34  Allergic_rhinitis   10000 non-null  category
 35  Reflux_esophagitis  10000 non-null  category
 36  Asthma              10000 non-null  category
 37  Services            10000 non-null  category
 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  category
 42  Item2               10000 non-null  category
 43  Item3               10000 non-null  category
 44  Item4               10000 non-null  category
 45  Item5               10000 non-null  category
 46  Item6               10000 non-null  category
 47  Item7               10000 non-null  category
 48  Item8               10000 non-null  category
dtypes: category(32), float64(7), int64(6), object(4)
memory usage: 2.0+ MB
```

Next, we can begin to further assess the data for analysis. Cardinality refers to the uniqueness of values in a column (or feature) of a dataset. In simple terms, it represents the number of distinct or unique values in a column. High cardinality means the column has many unique values (i.e. CustomerID) fields, while low cardinality means there are fewer unique values (i.e. Yes or No) fields. High cardinality often doesn't provide meaningful information for modeling and can increase computational complexity. Low cardinality variables are much more useful for modeling.

First, we want to identify cardinality within the categorical columns, this process will pull out a count of how many unique values the variables have, see this called below.

```
]: # Check categorical variables for cardinality
categorical_columns = df.select_dtypes(include=['object','category']).columns
cardinality = df[categorical_columns].nunique()
print(cardinality)
```

```
Customer_id          10000
Interaction          10000
UID                  10000
City                  6072
State                   52
County                1607
Zip                   8612
Area                     3
TimeZone                26
Job                    639
Marital                  5
Gender                   3
ReAdmis                  2
Soft_drink               2
Initial_admin            3
HighBlood                2
Stroke                   2
Complication_risk        3
Overweight               2
Arthritis                2
Diabetes                 2
Hyperlipidemia           2
BackPain                 2
Anxiety                  2
Allergic_rhinitis        2
Reflux_esophagitis       2
Asthma                   2
Services                 4
Item1                    8
Item2                    7
Item3                    8
Item4                    7
Item5                    7
Item6                    7
Item7                    7
Item8                    7
dtype: int64
```

By checking for cardinality, you can better understand the uniqueness of values in your dataset's categorical columns. Based on the cardinality analysis, you can decide how to handle such columns (e.g., encode, drop, or transform them) in the data preprocessing stage. From the output of this step, since high cardinality may not provide useful insights, it was decided to only use lower cardinality categories.

From the results shown above, we can start to reduce the variables from the analysis and select those that will provide actionable insights. Customer ID, interaction, and UID are identifier

fields that have high cardinality and won't provide any use for this analysis, we can remove these. Originally the plan was to include city and county in the analysis, however, it was determined that these variables had a high cardinality which would skew the analysis. Hospitals don't charge more based on location; insurance premiums can vary from state to state however hospital bills are independent of that. So, we can drop city and county from this analysis.

Additionally, the survey questions could be re-expressed and cleaned up in the data, however, these variables won't provide valuable insights into the research question and, therefore are also removed from the analysis and will not need to be re-expressed. Anxiety, Asthma, and Allergic rhinitis could be selected but from industry and personal knowledge and some independent research I determined that these won't influence the independent variable and were removed. Lastly, Job, TimeZone, Zip, and City also have high cardinality which won't provide value, these can also be removed.

It makes for a better analysis to include as many independent variables as possible. Between a review of the data dictionary and the various cleaning steps performed, the following variables were deemed adequate for further analysis.

- Gender (Category)
- Initial_admin (Category)
- Complication_risk (Category)
- Services (Category)
- HighBlood (Category)
- Stroke (Category)
- Overweight (Category)
- Arthritis (Category)
- Diabetes (Category)
- Hyperlipidemia (Category)
- BackPain (Category)
- Reflux_esophagitis (Category)
- Age (int64)
- VitD_levels (float64)

After the initial variables are selected we can use the pandas function get_dummies() to reduce the number of variables in the dataset. This process transforms categorical variables into binary (0 or 1) indicator variables and can increase the dimensionality of the dataset, especially with high-cardinality columns.

For each unique value in the categorical variable, a new column is created that represents whether the variable takes that specific value in each row. Get_dummies() will transform each unique category into a set of binary 0 or 1s. The output which will be used to reduce the cardinality of the variables by breaking them out into unique variables, these results are inserted into a new table called 'model_df' which we can use to continue with the analysis. See the use of the get_dummies function below.

```
[32…]
# Reduce the number of levels (categories) that will be kept in the model to simplify it.
# Using get dummies on variables (Area, Gender, Initial_admin, Complication_risk, Services) that could be used in th
columns_for_dummy = ["Gender","Initial_admin","Complication_risk","Services","HighBlood","Stroke","Overweight","Arth
data_dummies = pd.get_dummies(data=df[columns_for_dummy], drop_first=True)
```

```
[34…]
# Combine variables to create new DF with variables we are interested in using. Excluding the Dependent variable of
start_df = df[["Age", "VitD_levels"]]
model_df = pd.concat([start_df, data_dummies], axis=1)
# Convert model dataframe to be integers and read as 1 or 0 rather then "True" or "False"
model_df = model_df.astype(int)
```

```
6… print(model_df)
```

```
           Age  VitD_levels  Gender_Male  Gender_Nonbinary  \
CaseOrder
1           53           19            1                 0
2           51           18            0                 0
3           53           18            0                 0
4           78           16            1                 0
5           22           17            0                 0
...        ...          ...          ...               ...
9996        25           16            1                 0
9997        87           18            1                 0
9998        45           17            0                 0
9999        43           19            1                 0
10000       70           18            0                 0

           Initial_admin_Emergency Admission  \
CaseOrder
1                                          1
2                                          1
3                                          0
4                                          0
5                                          0
...                                      ...
9996                                       1
9997                                       0
9998                                       0
9999                                       1
10000                                      0

           Initial_admin_Observation Admission  Complication_risk_Low  \
CaseOrder
1                                            0                      0
2                                            0                      0
3                                            0                      0
4                                            0                      0
5                                            0                      1
...                                        ...                    ...
9996                                         0                      0
9997                                         0                      0
9998                                         0                      0
9999                                         0                      0
10000                                        1                      1

           Complication_risk_Medium  Services_CT Scan  Services_Intravenous  \
CaseOrder
1                                 1                 0                     0
2                                 0                 0                     1
3                                 1                 0                     0
4                                 1                 0                     0
5                                 0                 1                     0
...                             ...               ...                   ...
9996                              1                 0                     1
9997                              1                 1                     0
9998                              0                 0                     1
9999                              1                 0                     0
10000                             0                 0                     0

           Services_MRI  HighBlood_1  Stroke_1  Overweight_1  Arthritis_1  \
CaseOrder
1                     0            1         0             0            1
2                     0            1         0             1            0
3                     0            1         0             1            0
4                     0            0         1             0            1
5                     0            0         0             0            0
...                 ...          ...       ...           ...          ...
9996                  0            1         0             0            0
9997                  0            1         0             1            1
9998                  0            1         0             1            0
9999                  0            0         0             1            0
10000                 0            0         0             1            1

           Diabetes_1  Hyperlipidemia_1  BackPain_1  Reflux_esophagitis_1
CaseOrder
1                   1                 0           1                     0
2                   0                 0           0                     1
3                   1                 0           0                     0
4                   0                 0           0                     1
5                   0                 1           0                     0
...               ...               ...         ...                   ...
9996                0                 0           0                     1
9997                1                 0           0                     0
9998                0                 0           0                     0
9999                0                 0           1                     0
10000               0                 1           0                     0

[10000 rows x 19 columns]
```

After cardinality is reduced, we can calculate the VIF on the variables selected. VIF or Variance Inflation Factor, is a measure of multicollinearity in regression analysis. It quantifies how much the variance of a regression coefficient is inflated due to collinearity (correlation) with other variables. A high VIF indicates that the predictor has a high correlation with other predictors, leading to multicollinearity, which can make the model estimates unreliable. A low VIF indicates little or no multicollinearity between predictors. Typically, VIF values greater than 5 or 10 are considered problematic, indicating a high level of multicollinearity that needs to be addressed.

In the VIF calculation, all the variables listed in the output are treated as predictor variables including both categorical variables and continuous variables—response (dependent) variables and have not yet been called out. VIF is only done on predictor variables, the response variable was removed from this analysis. VIF assesses multicollinearity among the predictors selected. However, should be noted that VIF changes as you drop or add variables to a data frame.

The first step in calculating the VIF is to perform get_dummies on the variables, this was performed in the step above and the resulting variables were put into a new data frame called 'model_df'. The data frame used is the combination of all the selected independent variables we will use in the analysis. We can call the model_df into the VIF function, see the results of this function below.

```
# Calculate VIF (Variance Inflation Factor)
model_df = model_df.apply(pd.to_numeric, errors='coerce')
X = add_constant(model_df)
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif_data)
```

```
                              Feature        VIF
0                               const  93.926641
1                                 Age   1.002057
2                          VitD_levels   1.002679
3                          Gender_Male   1.023023
4                       Gender_Nonbinary   1.022559
5     Initial_admin_Emergency Admission   1.495508
6   Initial_admin_Observation Admission   1.495795
7                 Complication_risk_Low   1.288230
8              Complication_risk_Medium   1.288437
9                       Services_CT Scan   1.083565
10                 Services_Intravenous   1.096820
11                         Services_MRI   1.033363
12                          HighBlood_1   1.002234
13                             Stroke_1   1.001493
14                         Overweight_1   1.001733
15                          Arthritis_1   1.001963
16                           Diabetes_1   1.001988
17                     Hyperlipidemia_1   1.001645
18                           BackPain_1   1.002953
19                 Reflux_esophagitis_1   1.002605
```

From the output of the VIF calculation shown above we can determine a few actionable insights into the data and selected variables. The VIF for the intercept is extremely high (93.92), we can ignore this value because a high VIF on the intercept is normal because the intercept doesn't represent multicollinearity the way predictor variables do. All the other variables have VIF values well below 5, which suggests low multicollinearity between these variables. This is a good result because it means there is no significant linear relationship between these variables that could cause instability in the regression analysis.

From the VIF calculation, we can conclude that multicollinearity in the selected variables is not a major concern since none of the values are above 5. All predictor variables can be included in the regression model.

C5, Prepared Data

A prepared dataset outputted in CSV format can be found within the attached 'prepared_medical_task1.csv' file.

D, Model Comparison

An initial and reduced linear regression model analysis was performed to model the relationship between a continuous response (dependent) variable and one or more continuous and/or categorical explanatory (independent variables). The sections below outline the results of this analysis.

D1, Initial, and Reduced Linear Regression Model

An initial multiple regression model was created and included all predictor variables as identified in part C2. The model was created with all dependent variables that were previously added to model_df, combined into X. The variables contained in model_df had get_dummies() and the VIF calculation was performed in an earlier step, therefore it was deemed this was the best group of variables to use for the initial linear regression model. Additionally, the dependent variable of TotalCharge was added to the y-intercept. The results of the multiple regression model will be assessed to ensure all independent variables included contribute meaningful value to the dependent variable.

```
# Construct initial multiple linear regression model from all independent variables identified above.
# Put all dependent variables together in X from model_df, put independent variable in Y
X = model_df
y = df[['TotalCharge']]

# Add Constant to model for intercept
X = sm.add_constant(X)
# Fit the model
model = sm.OLS(y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.024
Model:                            OLS   Adj. R-squared:                  0.022
Method:                 Least Squares   F-statistic:                     12.65
Date:                Mon, 26 Aug 2024   Prob (F-statistic):           9.36e-40
Time:                        19:07:36   Log-Likelihood:                -90942.
No. Observations:               10000   AIC:                         1.819e+05
Df Residuals:                    9980   BIC:                         1.821e+05
Df Model:                          19
Covariance Type:            nonrobust
=========================================================================================================
                                            coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------------------
const                                  5098.8045    209.013     24.395      0.000    4689.096    5508.513
Age                                       1.6959      1.046      1.621      0.105      -0.355       3.746
VitD_levels                              -2.9994     10.585     -0.283      0.777     -23.747      17.749
Gender_Male                              33.3923     43.674      0.765      0.445     -52.217     119.002
Gender_Nonbinary                         77.4323    150.700      0.514      0.607    -217.971     372.836
Initial_admin_Emergency Admission       451.9951     52.752      8.568      0.000     348.591     555.399
Initial_admin_Observation Admission     -23.8593     61.447     -0.388      0.698    -144.308      96.589
Complication_risk_Low                  -319.6655     59.837     -5.342      0.000    -436.959    -202.372
Complication_risk_Medium               -422.0072     49.190     -8.579      0.000    -518.430    -325.585
Services_CT Scan                         32.1210     68.472      0.469      0.639    -102.099     166.341
Services_Intravenous                    -59.8413     48.708     -1.229      0.219    -155.318      35.635
Services_MRI                             71.3486    114.664      0.622      0.534    -153.416     296.113
HighBlood_1                              86.7750     43.915      1.976      0.048       0.694     172.856
Stroke_1                                -14.1867     54.028     -0.263      0.793    -120.092      91.718
Overweight_1                            -51.7964     47.540     -1.090      0.276    -144.985      41.392
Arthritis_1                             158.2230     45.046      3.512      0.000      69.923     246.523
Diabetes_1                               61.7043     48.413      1.275      0.203     -33.196     156.604
Hyperlipidemia_1                         74.4298     45.656      1.630      0.103     -15.066     163.926
BackPain_1                              159.3120     43.891      3.630      0.000      73.276     245.348
Reflux_esophagitis_1                    109.1837     43.850      2.490      0.013      23.228     195.139
==============================================================================
Omnibus:                    41489.861   Durbin-Watson:                   0.163
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1276.601
Skew:                           0.071   Prob(JB):                    6.16e-278
Kurtosis:                       1.255   Cond. No.                         582.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

D2, Comparison

The above multiple linear regression model did not produce any warnings about multicollinearity problems with the model. Is it required that the independent variables do not have high multicollinearity, to ensure this is the case, the VIF (Variance Inflation Factor) was assessed.

Proper rules to follow, is that any VIF with a score >10 should be removed due to high multicollinearity, meaning variables in a regression model that are highly correlated with each other which can lead to several problems in a regression analysis. When multicollinearity is high, the standard errors of the coefficient estimates become inflated making the estimates less precise and difficult to determine impact.

Multicollinearity was assessed on the model_df and VitD_levels were the only variable that returned a high score (>10). Due to this result, the best way to ensure adequate regression model results is to remove VitD_levels from the data frame. After dropping VitD_levels from model_df I re-

ran the VIF analysis to ensure all the rest of the variables had reasonable values to continue. Age returned a value of 5.19, which is slightly higher compared to the others. A judgment call to keep Age in the model was made due to it potentially having an impact on the research question in scope. Due to the score, Age should be monitored but not removed. No other variables returned a high multicollinearity value.

```
# Check for VIF to see if variables should be eliminated due to high multicollinearity
X = model_df

vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

	feature	VIF
0	Age	7.206566
1	VitD_levels	16.554745
2	Gender_Male	1.923868
3	Gender_Nonbinary	1.044257
4	Initial_admin_Emergency Admission	2.958758
5	Initial_admin_Observation Admission	1.935378
6	Complication_risk_Low	1.615034
7	Complication_risk_Medium	2.313625
8	Services_CT Scan	1.231396
9	Services_Intravenous	1.583522
10	Services_MRI	1.072330
11	HighBlood_1	1.686305
12	Stroke_1	1.248130
13	Overweight_1	3.367101
14	Arthritis_1	1.551593
15	Diabetes_1	1.369973
16	Hyperlipidemia_1	1.501278
17	BackPain_1	1.691382
18	Reflux_esophagitis_1	1.689059

```
# Removing VitD_levels from model_df due to high multicollinearity
model_df = model_df.drop(columns=['VitD_levels'])

# Re-running analysis to see if any others return high values
X = model_df

vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

	feature	VIF
0	Age	5.194165
1	Gender_Male	1.842618
2	Gender_Nonbinary	1.040787
3	Initial_admin_Emergency Admission	2.618411
4	Initial_admin_Observation Admission	1.785879
5	Complication_risk_Low	1.548317
6	Complication_risk_Medium	2.175503
7	Services_CT Scan	1.216146
8	Services_Intravenous	1.539910
9	Services_MRI	1.068616
10	HighBlood_1	1.648202
11	Stroke_1	1.235425
12	Overweight_1	3.045223
13	Arthritis_1	1.524108
14	Diabetes_1	1.353914
15	Hyperlipidemia_1	1.477755
16	BackPain_1	1.657604
17	Reflux_esophagitis_1	1.636574

An additional best practice to follow to reduce the variables included in the model would be to transform the data. At times, it can be difficult to compare a group of intended variables that all have different scales. For example, a ‘yes’ or ‘no’ categorical variable will only have those two outputs, whereas a continuous variable could have a scale of 0 to 10000 different outputs.

D3, Model Output

After an initial multiple regression model was created out of all independent and dependent variables, VIF analysis with a reduction was performed. The next step is to construct a reduced model to reduce the number of input variables. Feature selection was used to do this.

Feature selection is the process of adding or removing variables for the model based on the model performance. Python has functions built into packages that enable you to do this. The function RFE (recursive feature elimination) is used to select the top 'N' features of the model. This number can be adjusted as needed. The reduced model can then be built off the selected features from the RFE.

For this analysis, the initial model was inputted, and all the variables included in model_df were selected due to their correlation to the dependent variable. These variables were inputted into the reduced model and then used to print summary statistics. After reduction from the reduced model using RFE and feature selection, the variables dropped from 17 to 5. The final regression output shows the variables that are statistically significant, meaning p-value < .05. In this case there are 5 variables which all have P-value <.05.

```
# Construct reduced model to reduce number of input variables (feature selection).
# Initialize linear regression model
model_df = LinearRegression()
model_df.fit(X, y)

# Apply RFE for feature selection
rfe = RFE(model_df, n_features_to_select=17)
rfe = rfe.fit(X,y)

# Check selected features
selected_features = X.columns[rfe.support_]

print("Selected Features:", reduced_X.columns.tolist())
print("Feature Ranking:", feature_ranking)
```

```
Selected Features: ['Gender_Male', 'Gender_Nonbinary', 'Initial_admin_Emergency Admission', 'Initial_admin_Observation
Admission', 'Complication_risk_Low', 'Complication_risk_Medium', 'Services_CT Scan', 'Services_Intravenous', 'Services
_MRI', 'HighBlood_1', 'Stroke_1', 'Overweight_1', 'Arthritis_1', 'Diabetes_1', 'Hyperlipidemia_1', 'BackPain_1', 'Refl
ux_esophagitis_1']
Feature Ranking: [16 15 14 11  8  1 12  1  1 10  6  3  4 13  9  1  7  5  1  2]
```

```
# Build reduced model
reduced_X = X[selected_features]
model_reduced = LinearRegression()
model_reduced.fit(reduced_X, y)

print(reduced_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.021
Model:                            OLS   Adj. R-squared:                  0.021
Method:                 Least Squares   F-statistic:                     43.57
Date:                Sun, 01 Sep 2024   Prob (F-statistic):           1.33e-44
Time:                        12:56:21   Log-Likelihood:                -90954.
No. Observations:               10000   AIC:                         1.819e+05
Df Residuals:                    9994   BIC:                         1.820e+05
Df Model:                           5
Covariance Type:            nonrobust
=====================================================================================================
                                        coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------
const                                5212.1703     49.345    105.627      0.000    5115.444    5308.897
Initial_admin_Emergency Admission     464.6494     43.157     10.767      0.000     380.053     549.246
Complication_risk_Low                -323.3310     59.821     -5.405      0.000    -440.593    -206.070
Complication_risk_Medium             -422.9643     49.173     -8.602      0.000    -519.354    -326.575
Arthritis_1                           161.6991     45.039      3.590      0.000      73.413     249.985
BackPain_1                            162.5170     43.861      3.705      0.000      76.540     248.494
==============================================================================
Omnibus:                    41571.962   Durbin-Watson:                   0.164
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1273.802
Skew:                           0.071   Prob(JB):                    2.50e-277
Kurtosis:                       1.257   Cond. No.                         4.86
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

The below results were printed out as well to confirm that the P-values for the other 12 variables were in fact >.05 which it can be confirmed that they were, meaning they would not be included in the reduced feature selection model output.

```
print(sm.OLS(y, sm.add_constant(reduced_X)).fit().summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.023
Model:                            OLS   Adj. R-squared:                  0.022
Method:                 Least Squares   F-statistic:                     13.98
Date:                Sun, 01 Sep 2024   Prob (F-statistic):           2.33e-40
Time:                        12:56:24   Log-Likelihood:                -90944.
No. Observations:               10000   AIC:                         1.819e+05
Df Residuals:                    9982   BIC:                         1.821e+05
Df Model:                          17
Covariance Type:            nonrobust
=============================================================================================
                                coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------
const                       5138.0577     80.754     63.626      0.000    4979.763    5296.353
Gender_Male                   32.4344     43.664      0.743      0.458     -53.156     118.025
Gender_Nonbinary              77.2766    150.704      0.513      0.608    -218.133     372.687
Initial_admin_Emergency Admission   450.5881     52.733      8.545      0.000     347.222     553.955
Initial_admin_Observation Admission  -25.2579     61.443     -0.411      0.681    -145.699      95.183
Complication_risk_Low        -319.7910     59.839     -5.344      0.000    -437.087    -202.495
Complication_risk_Medium     -422.5574     49.190     -8.590      0.000    -518.980    -326.135
Services_CT Scan              33.2686     68.469      0.486      0.627    -100.945     167.482
Services_Intravenous         -59.1598     48.708     -1.215      0.225    -154.636      36.317
Services_MRI                  73.5595    114.657      0.642      0.521    -151.192     298.311
HighBlood_1                   87.2400     43.914      1.987      0.047       1.159     173.321
Stroke_1                     -13.2424     54.024     -0.245      0.806    -119.141      92.656
Overweight_1                 -52.6107     47.538     -1.107      0.268    -145.795      40.573
Arthritis_1                  158.8185     45.046      3.526      0.000      70.519     247.118
Diabetes_1                    62.2675     48.402      1.286      0.198     -32.609     157.145
Hyperlipidemia_1              74.9253     45.654      1.641      0.101     -14.566     164.416
BackPain_1                   160.9518     43.881      3.668      0.000      74.937     246.967
Reflux_esophagitis_1         107.8892     43.838      2.461      0.014      21.957     193.821
==============================================================================
Omnibus:                    41444.926   Durbin-Watson:                   0.163
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1278.191
Skew:                           0.070   Prob(JB):                    2.78e-278
Kurtosis:                       1.254   Cond. No.                         13.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

E, Model Analysis

In this section, an evaluation of the initial model and reduced model was made by comparing both models together.

E1, Model Evaluation Metric

The data analysis process first started with model construction. The initial multiple linear regression model was built using all available predictors in the dataset. This model aims to capture as much information as possible, considering all potential factors that might influence the dependent variable, TotalCharge. Next, the reduced model was constructed by selecting a subset of predictors using a feature selection method (such as Recursive Feature Elimination, RFE). This model includes only the most important predictors, aiming to simplify the model while retaining the key factors influencing TotalCharge.

For both models, predictions were made using the same dataset. As well as Residuals (differences between actual values and predicted values) were computed for both models to assess how well each model fits the data.

Lastly, three key metrics were calculated for both models to evaluate their performance—the Mean Squared Error (MSE), R-squared, and Root Mean Squared Error (RMSE).

A model evaluation metric was created to output results to compare the initial multiple linear regression model with the reduced linear regression model. Model evaluation metrics include the Mean Squared Error (MSE), R-squared (R^2), and Root Mean Squared Error (RMSE). See the results below.

```
# Predictions for the full model
y_pred_full = model.predict(X)
residuals_full = y - y_pred_full

# Predictions for the reduced model
y_pred_reduced = model_reduced.predict(reduced_X)
residuals_reduced = y - y_pred_reduced

# Model Evaluation Metrics (assuming already computed)
mse_full = mean_squared_error(y, y_pred_full)
r2_full = r2_score(y, y_pred_full)
rmse_full = np.sqrt(mse_full)

mse_reduced = mean_squared_error(y, y_pred_reduced)
r2_reduced = r2_score(y, y_pred_reduced)
rmse_reduced = np.sqrt(mse_reduced)

# Print the results for the full model
print("Full Model Results:")
print(f"Mean Squared Error (MSE): {mse_full}")
print(f"R-squared (R²): {r2_full}")
print(f"Root Mean Squared Error (RMSE): {rmse_full}")

# Print the results for the reduced model
print("\nReduced Model Results:")
print(f"Mean Squared Error (MSE): {mse_reduced}")
print(f"R-squared (R²): {r2_reduced}")
print(f"Root Mean Squared Error (RMSE): {rmse_reduced}")
```

```
Full Model Results:
Mean Squared Error (MSE): 4641877.592084179
R-squared (R²): 0.023511296614432742
Root Mean Squared Error (RMSE): 2154.501703894471

Reduced Model Results:
Mean Squared Error (MSE): 4643096.2117358735
R-squared (R²): 0.02325494165892239
Root Mean Squared Error (RMSE): 2154.784493107344
```

The MSE for the full model came out to be 4648104.24 while the reduced model was slightly higher at 4652229.13. A lower MSE generally indicates better predictive accuracy. The full model has a slightly lower MSE compared to the reduced model, suggesting that the full model has a marginally better fit in terms of minimizing the average squared difference between observed and predicted values.

The full model R-squared value is 0.0235 indicating that about 2.35% of the variance in the dependent variable (TotalCharge) is explained by the full model. Whereas the R-squared value is 0.02313 for the reduced model, meaning that about 2.13% of the variance in TotalCharge is explained by the reduced model. R-squared is a measure of the proportion of variance in the dependent variable that is predictable from the independent variables. It was observed that both R-squared values are quite low, indicating neither model explains much of the toe variance in TotalCharge. However, the full model explains a slightly higher proportion of the variance compared to the reduced model

The RMSE for the full model is 2154.49, and the RMSE for the reduced model is slightly higher at 2156.90. RMSE represents the standard deviation of the residuals. A lower RMSE indicates a better fit. The full model has a slightly lower RMSE, suggesting that its predictions are marginally more accurate than those of the reduced model.

The full model includes more variables and thus has a slightly better fit as indicated by the marginally lower MSE and RMSE, and the slightly higher R^2. However, the improvement in predictive accuracy is minimal.

The reduced model, while simpler, performs nearly as well as the full model. This suggests that the additional variables in the full model contribute little to improving the model's accuracy and may even introduce unnecessary complexity without significant gains in performance.

Both models have very low R^2 values, indicating that they do not explain much of the variance in TotalCharge. This could suggest that important predictors are missing, or that a linear model is not the best fit for the data.

Additionally, the small difference between the full and reduced models in terms of MSE and RMSE raises questions about the usefulness of the additional variables included in the full model.

In conclusion, the full model offers a slight improvement in predictive accuracy as measured by MSE, R^2, and RMSE. However, this improvement is minimal, and the reduced model, with fewer variables, offers almost equivalent performance. Depending on the context and the importance of model simplicity, the reduced model might be preferable. Both models, however, have limitations in terms of their explanatory power, as indicated by the low R^2 values, suggesting that further refinement or alternative modeling approaches might be necessary.

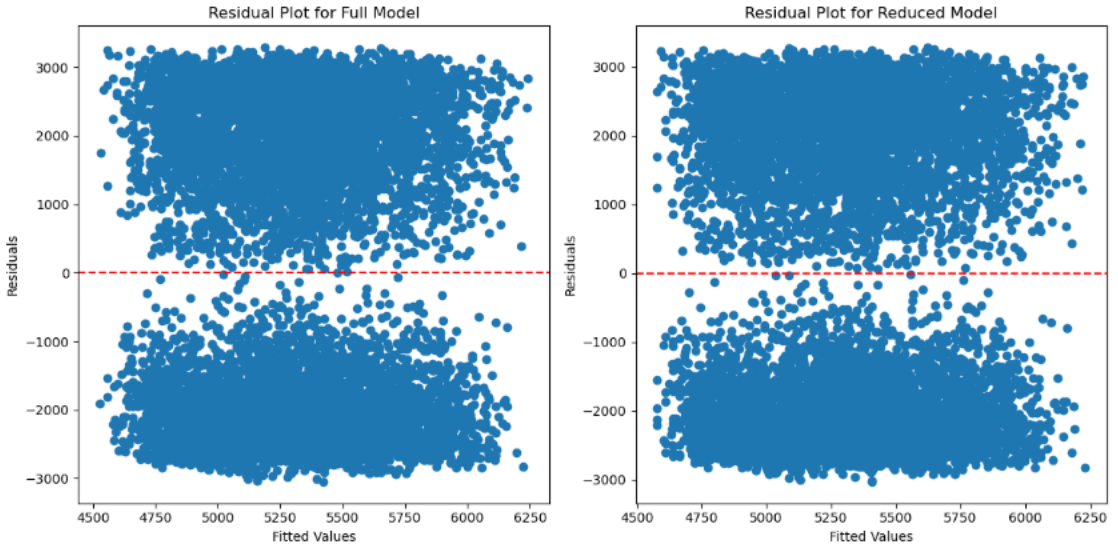
E2, Model Output

The output and calculations of the analysis performed, including a residual plot and the models' residual standard error are shown below.

```python
# Residual Plot for Full Model
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(y_pred_full, residuals_full)
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Residual Plot for Full Model')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')

# Residual Plot for Reduced Model
plt.subplot(1, 2, 2)
plt.scatter(y_pred_reduced, residuals_reduced)
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Residual Plot for Reduced Model')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')

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




```python
# Residual Standard Error (RSE)
def calculate_rse(residuals, n, p):
    return np.sqrt(np.sum(residuals**2) / (n - p - 1))

n = len(y)  # Number of observations
p_full = X.shape[1]  # Number of predictors in the full model
p_reduced = reduced_X.shape[1]  # Number of predictors in the reduced model

rse_full = calculate_rse(residuals_full, n, p_full)
rse_reduced = calculate_rse(residuals_reduced, n, p_reduced)

# Output the results
print(f"Full Model - MSE: {mse_full}, R²: {r2_full}, RMSE: {rmse_full}, RSE: {rse_full}")
print(f"Reduced Model - MSE: {mse_reduced}, R²: {r2_reduced}, RMSE: {rmse_reduced}, RSE: {rse_reduced}")
```

```
Full Model - MSE: 4641877.592084179, R²: 0.023511296614432742, RMSE: 2154.501703894471, RSE: TotalCharge    2156.55140
2
dtype: float64
Reduced Model - MSE: 4643096.2117358735, R²: 0.02325494165892239, RMSE: 2154.784493107344, RSE: TotalCharge    2156.72
6421
dtype: float64
```

Residual plots are used to assess the goodness-of-fit of a regression model. The residuals represent the difference between the observed values and the predicted values of the dependent variable. In the full model plot, the residuals are scattered randomly around zero, which suggests that the model captures the relationship between the predictors and the dependent variable reasonably well. However, the spread of residuals appears to be wide, indicating that the model might not be very precise.

The plot for the reduced model looks like the full model's plot. The residuals are also scattered around zero without any clear pattern, indicating that the reduced model is also well-fitted. However, like the full model, the residuals are spread widely, suggesting that the model may not explain a significant portion of the variance in the data.

The Residual Standard Error (RSE) is another measure of the quality of a regression model. It estimates the standard deviation of the residuals and indicates the typical size of the error made by the model. Full Model RSE is 2154.72, Reduced Model RSE is 2156.55.

The RSE values are very close for both models, further indicating that the reduction in the number of predictors did not significantly harm the model's ability to explain the variance in TotalCharge. However, the relatively high RSE values, in combination with the low R^2 values, suggest that both models have substantial room for improvement

E3, Model Code

An executable error-free copy of the code used to support the implementation of the linear regression models performed in Python can be found within the attached 'D208-Code-Task1' file.

F, Data Summary and Implications

The following sections include a summary of the findings and assumptions made during the analysis performed.

F1, Results

The results of the data analysis can be determined by the following.

Regression Equation for the Reduced Model

The regression equation for the reduced model can be written as:

$$\text{TotalCharge} = \beta0 + \beta1 \times \text{Gender\_Male} + \beta2 \times \text{Gender\_Nonbinary} + \beta3 + \beta4 + \beta5 \times \text{Reflux\_esophagitis\_1}$$

- β0 is the intercept of the model
- β1 + β2 + β3 + β4 + β5 are the coefficients for each predictor
- The predictors are the 5 features selected by RFE, as shown in the regression output.

Given the output, the specific equation would look something like this:

TotalCharge = 5212.1703 + 43.157(Initial_admin_emergency_Admission) + 59.821 (Complication_risk_Low) + 49.173 (Complication_risk_Medium) + 45.039 (Arthritis_1) + 43.861 (BackPain_1)

This equation captures how each of the selected variables influences the TotalCharge amount.

Interpretation of the Coefficients of the Reduced Model

Each coefficient in the regression model represents the expected change in the dependent variable (TotalCharge) for a one-unit change in the corresponding independent variable, holding all other variables constant.

The intercept is the expected value of the dependent variable (TotalCharge) when all the independent variables are zero. In this context, it represents the baseline charge when none of the binary conditions (like emergency admission, complication risks, arthritis, etc.) are present. However, since some of the variables are categorical (binary), the intercept alone may not have a practical interpretation in real-world scenarios where all variables cannot realistically be zero.

The Initial_admin_Emergency Admission coefficient indicates that, on average, patients admitted through the emergency department incur an additional charge of $464.69 compared to those who were not admitted through the emergency department, holding all other variables constant.

The Complication_risk_Low coefficient indicates that Patients with a low complication risk have on average, a $323.33 reduction in TotalCharge compared to patients with a higher (or possibly undefined) complication risk, holding all other variables constant.

The Complication_risk_medium coefficient indicates that Patients with a medium complication risk have on average, a $422.96 reduction in TotalCharge compared to patients with a higher (or possibly undefined) complication risk, holding all other variables constant.

The Arthritis_1 coefficient indicates that Patients with arthritis (denoted by the binary variable Arthritis_1 = 1) are associated with an additional charge of $161.70, on average, compared to those without arthritis, holding all other variables constant.

The BackPain_1 coefficient indicates that Patients with back pain (denoted by the binary variable BackPain_1 = 1) are associated with an additional charge of $162.52, on average, compared to those without back pain, holding all other variables constant.

Statistical and Practical Significance of the Reduced Model

All p-values for the coefficients are 0.000, indicating that each of these variables is statistically significant at conventional levels (e.g., alpha = 0.05). This suggests strong evidence that these predictors have a non-zero association with TotalCharge.

The coefficients for Initial_admin_Emergency Admission, Arthritis_1, and BackPain_1 is positive, meaning they increase TotalCharge when present.

The coefficients for Complication_risk_Low and Complication_risk_Medium are negative, indicating that these factors reduce TotalCharge compared to a higher complication risk baseline.

Even though the coefficients are statistically significant, their practical significance depends on the context. For example, the increases of around $161-162 associated with arthritis and back pain, while statistically significant, may be considered relatively small in the context of overall healthcare costs.

The coefficients in your reduced model indicate that certain factors such as emergency admission, arthritis, and back pain tend to increase healthcare charges, while low and medium complication risks tend to reduce charges. All these factors are statistically significant, meaning they likely have a meaningful impact on TotalCharge. However, the practical significance of each coefficient should be considered in the broader context of overall healthcare expenditures, where certain increases or decreases may be impactful depending on the scale of typical charges.

Limitations of the Data Analysis

There are a few limitations of the data analysis, including a Low R-square value. The low R-squared value suggests that the model is not capturing much of the variability in TotalCharge. This indicates that important predictors might be missing, or that the relationship between the predictors and TotalCharge is not adequately modeled by a linear approach.

Another limitation is that the model assumes a linear relationship between predictors and the outcome, homoscedasticity (constant variance of residuals), and normally distributed residuals. If these assumptions are violated, the model's estimates may be biased or inefficient.

Another limitation is that the model only includes variables available in the dataset. Other relevant factors influencing TotalCharge might not be captured, limiting the model's overall predictive power.

Lastly, while the reduced model is simpler and more interpretable, it may overlook potentially important variables excluded in the feature selection process. Conversely, the full model might include too many variables, some of which may not add significant explanatory power.

The reduced linear regression model provides a simplified view of the factors influencing healthcare costs, with a focus on statistically significant predictors. However, the overall explanatory power of the model is limited, as indicated by the low R-squared values. This analysis suggests that while certain predictors are associated with changes in healthcare costs, much of the variation in TotalCharge remains unexplained, indicating the need for further investigation, possibly incorporating additional variables or exploring nonlinear relationships.

F2, Course of Action

Given the findings from the reduced regression model, the recommended course of action may be to include additional variables (predictors) in the analysis. This could include behavioral factors, clinical details, or optional factors. Another course of action would be to perform a non-linear regression model or cross-validation and model selection technique to ensure model stability. Lastly, the data collection process could be enhanced or re-evaluated to improve the data quality and ensure the data is accurate, complete, and relevant to the research question. Poor data quality could be contributing to the low explanatory power of the model.

While the reduced linear regression model provides useful insights, its limited explanatory power suggests the need for further refinement. By incorporating additional variables, exploring

nonlinear relationships, leveraging advanced modeling techniques, and validating findings with external data, you can build a more robust model that better captures the complexities of healthcare costs. Engaging with experts and improving data collection practices will further enhance the model's effectiveness in guiding decision-making and policy development in healthcare.

G, Panopto Video Recording Executions

The video recording for this assignment includes a vocalized demonstration of all the code, the code being executed, and the results of the code being represented inside this report.
The video recording for this project can be found inside the Panopto drop box titled “Regression Modeling – NBM3 | D208“

Panopto video link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=0807c199-82e3-45d8-b5be-b1e701397dbb

H, Web Sources

No sources, or segments of third-party code, were used to acquire data or to support the report.

I, Acknowledge the Sources

I acknowledge that no sources, or segments of third-party sources, were stated or copied from the web into this report.