Predictive Modeling Task 1

D208 WGU

TRESA (TESSIE) AUSTIN

Part I: Research Question

A. Define the research question and describe the purpose of this data analysis:

1. Patients that enter the hospital for whatever reason, want to get well and go home. Whether there because of an emergency, a planned surgery or even a joyful event such as having a child, the goal is to not be in the hospital any longer than needed. Insurance companies also want hospital stays to be as short as possible since it's costly to care for patients in that setting. Based on that it's important to research the following question:
   "What factors can predict the total charges a patient will incur during their initial hospital stay?"

2. Define the goals of the data analysis.
   If we can determine what predicates how long a patient is initially in the hospital, then we can determine how to treat patients outside the hospital and/or shorten their stay in the hospital. Insurance companies can then determine how to spend monies to improve preventive care and other options for care outside of a costly hospital stay.

Part II: Method Justification

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

1. Summarize **four** assumptions of a multiple linear regression model. Bobbitt (2021) advises that multiple linear regression is a form of statistical analysis that can be used to determine and understand the relationships between multiple predictor variables and a response variable. To perform a multiple linear regression we have to make some assumptions. These are listed below.
   a. Linearity implies that the relationship between a dependent variable and each independent variable should be linear. A change in the dependent variable should be in direct alignment or proportionality to the change in the dependent variable.
   b. Normally distributed residuals should occur. There are various tests that can be used to determine if the residuals are distributed as they should be. Using a goodness of fit test will advise if the data is normal . If the data is not normal then effects of multicollinearity could be introduced
   c. Multicollinearity should not exist. Independent variables should not be too tightly correlated with each other. If this is found to exists it can be hard to determine which variable explains the shared variance with the outcome or it may suggest that there are two variables responsible for the same underlying factor (NCRM, 2011)
   d. Homoscedasticity should occur meaning the variance of the residuals should be the same at each level of the variable (NCRM, 2011). Using a scatterplot will assist in seeing how the data aligns.
2. Describe **two** benefits of using Python in support of various phases of the analysis.
   a. Python allows the use of a wide variety of data manipulation using lists, tuples, dictionaries, and sets according to BeyondVerse (2023). The article goes on to say that the use of lists allows versatility, the use of tuples can hold various items but are immutable to ensure data integrity, dictionaries compile collections of values in keyed pairs for efficient data retrieval and sets help to determine and eliminate duplicates since they are unordered collections. These options allow a user to structure and manipulate data easily.
   b. Python also boasts a large variety of libraries for advanced linear regression. According to Luca (2016) Python offers a mature system of packages for data analysis that are very versatile. There are specific packages for commonly used optimization algorithms such as Scikit-learn. Scikit-learn is built on top of SciPy and offers modules for data processing, model selection and validation for data analysis. As a software engineer, Python has translatable syntax and debugging as noted on WGU Information Technology page (nd)

3. Explain why multiple linear regression an appropriate technique is to use for analyzing our research question

The use of multiple linear regression to explore the research question "What factors can predict the total charges a patient will incur during their initial hospital stay?" helps to assess the association between two or more independent variables and a single continuous dependent variable. For the data that will be used, independent variables such as whether patients have high blood pressure, diabetes, or if they have had a stroke, and their complication risk level can come into play. The dependent variable of the number of total charges a patient incurred while in the hospital can be looked at through the lens of the multiple independent variables to reach a conclusion.

If we identify an independent variable as a confounder then we can use multiple linear regression to estimate the association between that risk and the outcome according to an article from Boston University School of Public Health(2013). The objective here is to use known values of independent variables to predict the value of the dependent variable.

Part III: Data Preparation

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

1. Describe your data cleaning goals and the steps used to clean the data

To complete the needed data analysis, the provided file will need to be cleaned to prepare for use.

- An initial check for duplicates and missing values will be done.
- After that inspection of the data types and variables with data will be done.
    - Several data types will need to be changed from their current values to ones that are applicable for multiple regression analysis.
        - There are columns that will need to be changed from True/False (true/false) to 1/0 numeric values.
        - Other columns that are of nominal categorical type will need to be adjusted to represent data in a binary numeric form using one hot encoding as described in D208 – Webinar: Getting Started with D208 Part 1 (WGU, 2022). The use of pandas get_dummies() is the best option for completing the needed transformation of data for this project as it takes a series or list and converts each element present to a column heading as it iterates over the data and inserts values of 1 if it's included and 0 if not (Waqar, 2024).

2. Describe the dependent variable and *all* independent variables using summary statistics required to answer the research question "What factors can predict the total charges a patient will incur during their initial hospital stay?"

Dependent Variable:

- TotalCharge
    - The amount charged to the patient daily. This value reflects an average per patient based on the total charge divided by the number of days hospitalized. This amount reflects the typical charges billed to patients, not including specialized treatments.

```
Summary Statistics for TotalCharge:
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
```

Total Charges for a patient can vary based on treatment. The above calculations are found by dividing th e total amount billed by the number of days hospitalized.

Independent Variables:

- Children
  - Number of children in the patient's household as reported in the admissions information (might not be children of the patient)

Number of Children	Count of Patients	Percentage of patients
0	2548	25.48%
1	2509	25.09%
2	1475	14.75%
3	1489	14.89%
4	995	9.95%
5	169	1.69%
6	191	1.91%
7	213	2.13%
8	209	2.09%
9	108	1.08%
10	94	0.94%

Name: count, dtype: int64

Self-reported numbers at the time of admission can have issues with reliability. This number represents the number of children in the household that may or may not be related to the patient. Approximately 25% of patients reported no children in the home and another 25% reported one child in the home.

- Age:
  - Age of the patient as reported in admissions information

Age counts:

18 133
19 137
20 120
21 125
22 141
...
85 135
86 156

87 136
88 143
89 132
Name: count, Length: 72, dtype: int64

While not all ages are shown here, what we do see is the youngest patient is 18. There are no children in the data set. The oldest patient is 89. There appear to be about 1 percent of the total population in the data set aligned with each age.

- Income:
  - Annual income of the patient (or primary insurance holder) as reported at time of admission

Income counts:
154.08 1
300.79 1
395.23 1
401.86 1
493.04 1
..
197576.18 1
197675.00 1
203774.60 1
204542.41 1
207249.10 1
Name: count, Length: 9993, dtype: int64

Income is self-reported when the patient is being admitted. Again while not all values are shown here, we can see that the reported values range from about $150 a year to over $200,000 per year. This is quite a disparity in income levels.

- VitD_levels:
  - The patient's vitamin D levels as measured in ng/mL

VitD_levels counts:
9.806483 1
10.315234 1
10.877427 1
11.083430 1
11.475314 1
..
24.637420 1
24.889110 1
25.147270 1
25.444099 1
26.394449 1
Name: count, Length: 9976, dtype: int64

There is a wide range of Vitamin D levels among patients, with very few patients having the exa ct same vitamin D level. This value can indicate major issues with patient health.

- Doc_visits:
  - Number of times the primary physician visited the patient during the initial hospitalization

Doc_visits counts:
1 6
2 58
3 595
4 2385
5 3823

6 2436
7 634
8 61
9 2
Name: count, dtype: int64

Visits by the primary physician for each patient ranged from 1 to 9 times. With most patients s eeing their doctor between 4-6 times.

- Full_meals_eaten:
  - Number of full meals the patient ate while hospitalized (partial meals count as 0, and some p atients had more than three meals in a day if requested)

Full_meals_eaten counts:
0 3715
1 3615
2 1856
3 612
4 169
5 25
6 6
7 2
Name: count, dtype: int64

The number of full meals eaten while in the hospital can indicate how well a patient is recovering. If a patient is not eating that could mean they don't feel well or aren't physically able to eat.

- HighBlood:
  - Whether the patient has high blood pressure (True, False)

HighBlood counts:
No 5910
Yes 4090
Name: count, dtype: int64

Approximately 59% of patients reported had high blood pressure, while the other almost 41% reported not having high blood pressure.

- Stroke:
  - Whether the patient has had a stroke (True, False)

Stroke counts:
No 8007
Yes 1993
Name: count, dtype: int64

For stroke, we see about 20% of patients have had a stroke and approximately 80% have not.

- Diabetes:
  - Whether the patient has diabetes (True, False)

Diabetes counts:
No 7262
Yes 2738
Name: count, dtype: int64

Of all patients, 27% were diabetic and 73% were not.

- Additional_charges:
    - The average amount charged to the patient for miscellaneous procedures, treatments, medicines, anesthesiology, etc.

  Additional_charges counts:
  3125.703000 1
  3132.259990 2
  3139.049369 2
  3173.112679 1
  3213.079900 1
  ..
  30087.650940 1
  30395.025240 1
  30422.530000 1
  30466.930000 1
  30566.070000 1
  Name: count, Length: 9418, dtype: int64
- Initial_admin
    - The means by which the patient was admitted into the hospital initially (emergency admissi on, elective admission, observation)

  Initial_admin counts:
  Elective Admission 2504
  Emergency Admission 5060
  Observation Admission 2436
  Name: count, dtype: int64

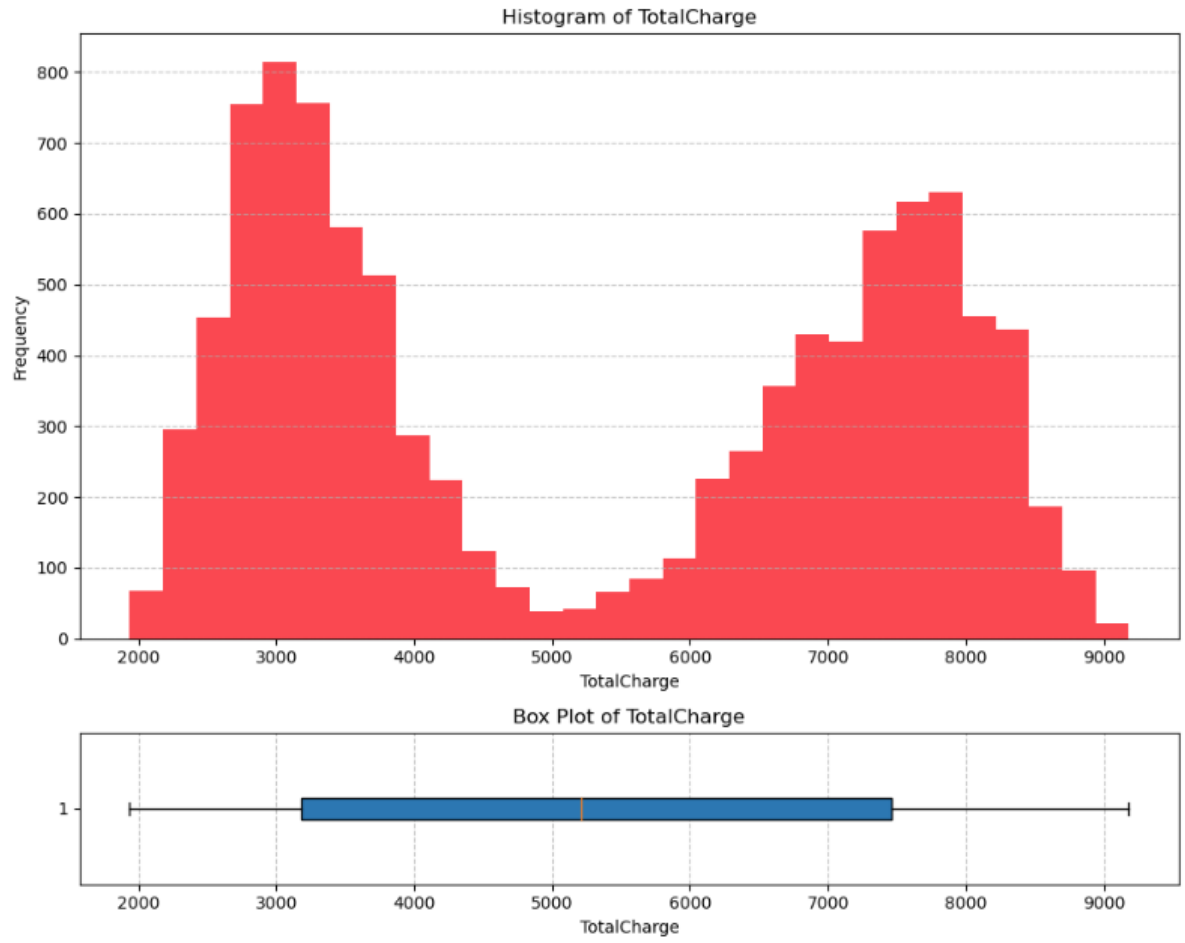
There are three categories of admission for patients. Approximately 50% are admitted via emergency procedures with another 25% being admitted for elective procedures and almost 25% for observation.

3. Generate univariate and bivariate visualizations of the distributions of the dependent and independent variables, including the dependent variable in your bivariate visualizations.

   For our dependent variable two types of visualizations were created. First a histogram to show the shape of the distribution, frequency distribution and variability of the data within this variable. According to Sharma (2019) the x-axis denotes the number of bins while the y-axis provides the frequency. The number of bins are a parameter that can be modified to update how you want to visualize the distribution in a histogram.
   Secondly a box plot which provides some of the same visualizations on distribution and variability, but in a way that can better define quartiles and show outliers. Box plots are effective at presenting outliers in a way that is easy for users to see. These outliers are shown as data points outside the "whiskers" of the plot making them easy to identify (Statistics By Jim, 2021).

```
# Total Charges Univariate and Bivariate

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8), gridspec_kw={'height_ratios': [4, 1]})
```

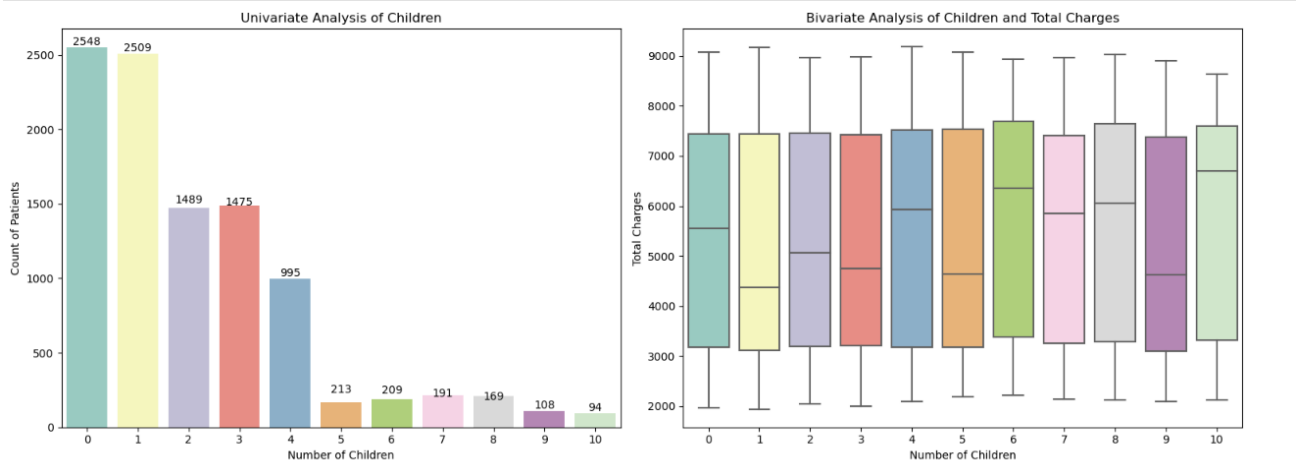
```
# Histogram for Total Charges
ax1.hist(df["TotalCharge"], bins=30, color='red', alpha=0.7)
ax1.set_title('Histogram of TotalCharge')
ax1.set_xlabel('TotalCharge')
ax1.set_ylabel('Frequency')
ax1.grid(axis='y', linestyle='--', alpha=0.7)

# Box plot for Total Charges
ax2.boxplot(df["TotalCharge"], vert=False, patch_artist=True)
ax2.set_title('Box Plot of TotalCharge')
ax2.set_xlabel('TotalCharge')
ax2.grid(axis='x', linestyle='--', alpha=0.7)

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



Additional visualizations of all variables was conducted, code and graphs are below.

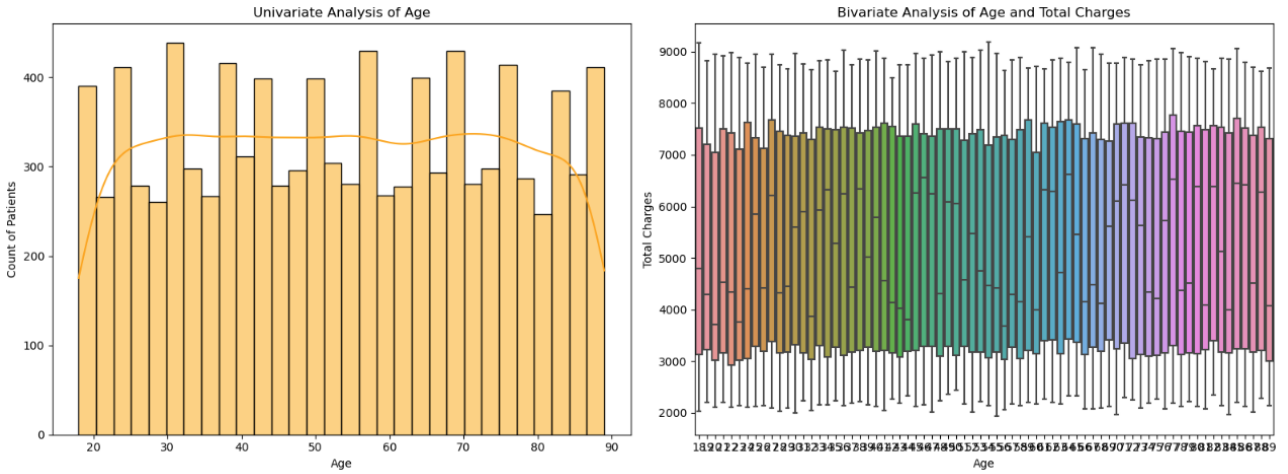
```
# Children Univariate and Bivariate

Children_counts = df["Children"].value_counts()


fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))


# Bar chart for # Children

sns.barplot(x=Children_counts.index, y=Children_counts.values, palette="Set3", ax=ax1)

ax1.set_title('Univariate Analysis of Children')

ax1.set_xlabel('Number of Children')
```

```
ax1.set_ylabel('Count of Patients')
for index, value in enumerate(Children_counts.values):
    ax1.text(index, value, str(value), ha='center', va='bottom')

# Box plot for Children and Total Charges
sns.boxplot(x=df["Children"], y=df["TotalCharge"], ax=ax2, palette="Set3")
ax2.set_title('Bivariate Analysis of Children and Total Charges')
ax2.set_xlabel('Number of Children')
ax2.set_ylabel('Total Charges')

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

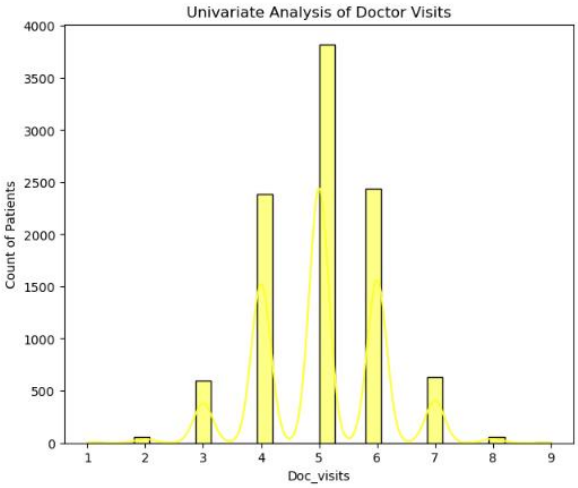


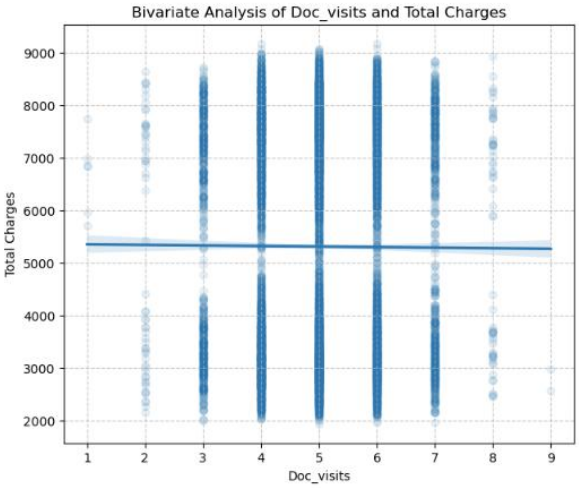

```
# Age Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Histogram for Age
sns.histplot(df["Age"], kde=True, bins=30, color="orange", ax=ax1)
ax1.set_title('Univariate Analysis of Age')
ax1.set_xlabel('Age')
ax1.set_ylabel('Count of Patients')

# Box plot for Age and Total Charges
sns.boxplot(x=df["Age"], y=df["TotalCharge"], ax=ax2)
ax2.set_title('Bivariate Analysis of Age and Total Charges')
ax2.set_xlabel('Age')
```

```
ax2.set_ylabel('Total Charges')

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

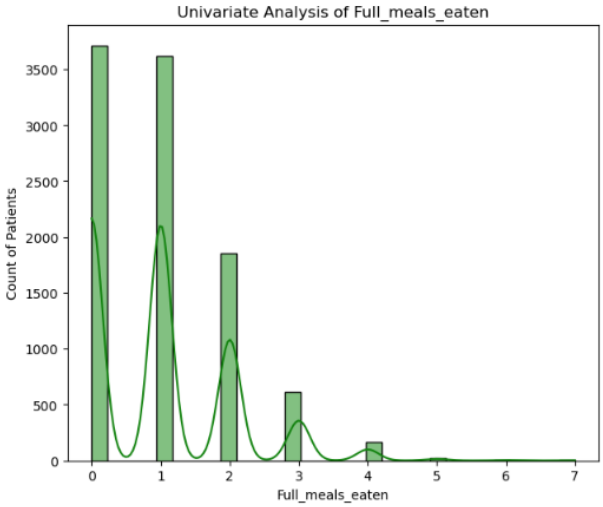


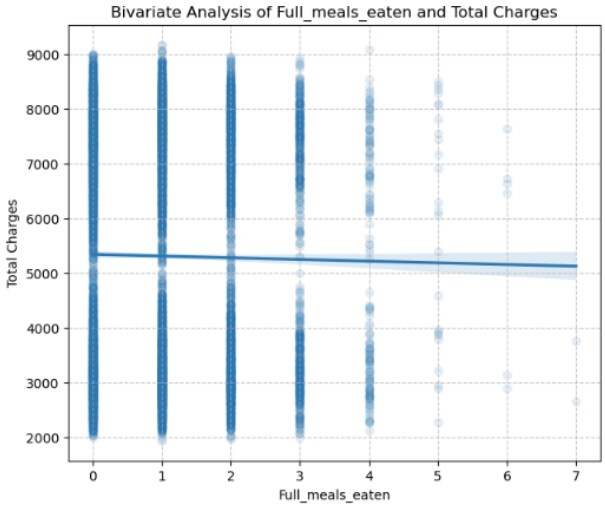

```
# Doc_visits Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Graph for Doc_visits
sns.histplot(df["Doc_visits"], kde=True, bins=30, color="yellow", ax=ax1)
ax1.set_title('Univariate Analysis of Doctor Visits')
ax1.set_xlabel('Doc_visits')
ax1.set_ylabel('Count of Patients')

# Graph for Doc_visits and TotalCharge
plt.title("Bivariate Analysis of Doc_visits and Total Charges")
sns.regplot(data=df, x="Doc_visits", y="TotalCharge", scatter_kws={'alpha': 1/10})
plt.xlabel('Doc_visits')
plt.ylabel('Total Charges')
plt.grid(axis='both', linestyle='--', alpha=0.7)
plt.show()
```





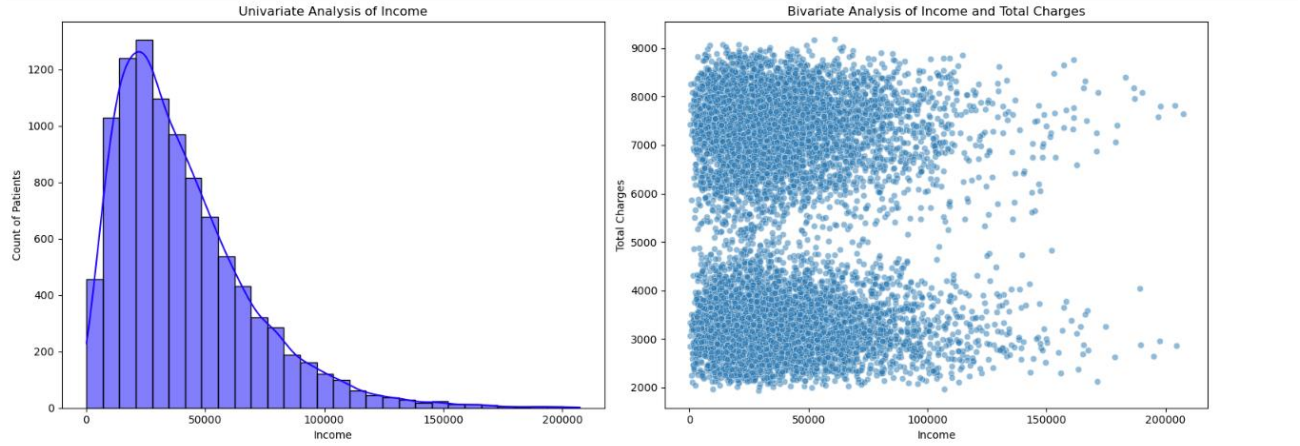

```
# Full_meals_eaten Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Graph for Full_meals_eaten
sns.histplot(df["Full_meals_eaten"], kde=True, bins=30, color="green", ax=ax1)
ax1.set_title('Univariate Analysis of Full_meals_eaten')
ax1.set_xlabel('Full_meals_eaten')
ax1.set_ylabel('Count of Patients')

# Graph for Full_meals_eaten and TotalCharge
plt.title("Bivariate Analysis of Full_meals_eaten and Total Charges")
sns.regplot(data=df, x="Full_meals_eaten", y="TotalCharge", scatter_kws={'alpha': 1/10})
plt.xlabel('Full_meals_eaten')
plt.ylabel('Total Charges')
plt.grid(axis='both', linestyle='--', alpha=0.7)
plt.show()
```





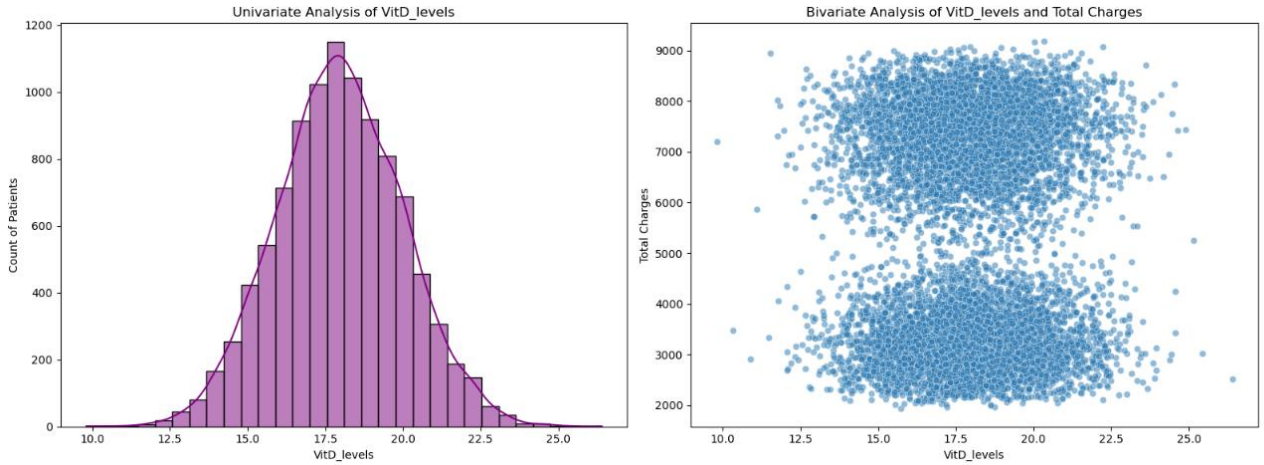

```
# Income Univariate and Bivariate
```

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Histogram for Income
sns.histplot(df["Income"], kde=True, bins=30, color='blue', ax=ax1)
ax1.set_title('Univariate Analysis of Income')
ax1.set_xlabel('Income')
ax1.set_ylabel('Count of Patients')

# Scatterplot of Income and Total Charges
sns.scatterplot(x=df["Income"], y=df["TotalCharge"], ax=ax2, alpha=0.5)
ax2.set_title('Bivariate Analysis of Income and Total Charges')
ax2.set_xlabel('Income')
ax2.set_ylabel('Total Charges')

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

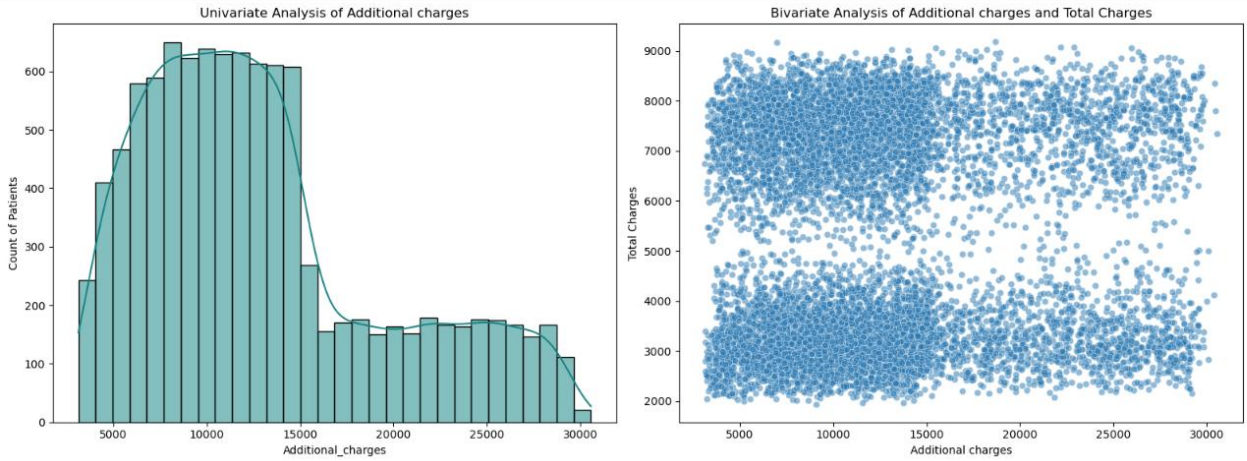



```
# VitD_levels Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Histogram for VitD_levels
sns.histplot(df["VitD_levels"], kde=True, bins=30, color="purple", ax=ax1)
ax1.set_title('Univariate Analysis of VitD_levels')
ax1.set_xlabel('VitD_levels')
ax1.set_ylabel('Count of Patients')

# Scatterplot of Income and Total Charges
sns.scatterplot(x=df["VitD_levels"], y=df["TotalCharge"], ax=ax2, alpha=0.5)
ax2.set_title('Bivariate Analysis of VitD_levels and Total Charges')
ax2.set_xlabel('VitD_levels')
ax2.set_ylabel('Total Charges')

plt.tight_layout()
```

```
plt.show()
```

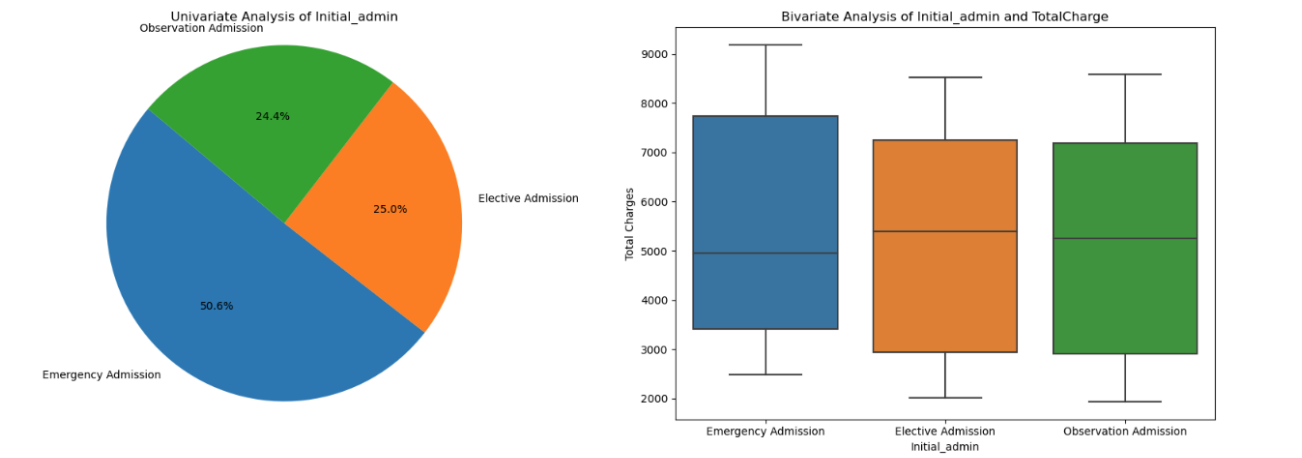



```
# Additional_charges Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Histogram for Additional_charges
sns.histplot(df["Additional_charges"], kde=True, bins=30, color='teal', ax=ax1)
ax1.set_title('Univariate Analysis of Additional charges')
ax1.set_xlabel('Additional_charges')
ax1.set_ylabel('Count of Patients')

# Scatterplot of Additional_charges and Total Additional_charges
sns.scatterplot(x=df["Additional_charges"], y=df["TotalCharge"], ax=ax2, alpha=0.5)
ax2.set_title('Bivariate Analysis of Additional charges and Total Charges')
ax2.set_xlabel('Additional charges')
ax2.set_ylabel('Total Charges')

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



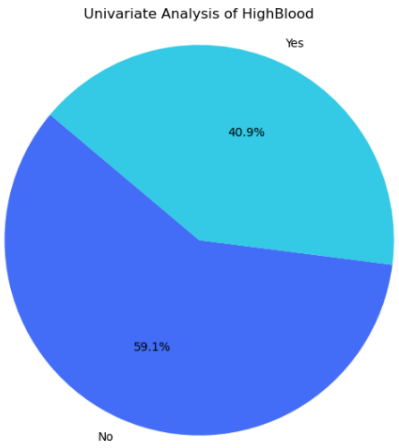

```
# Initial_Admin Univariate and Bivariate
```

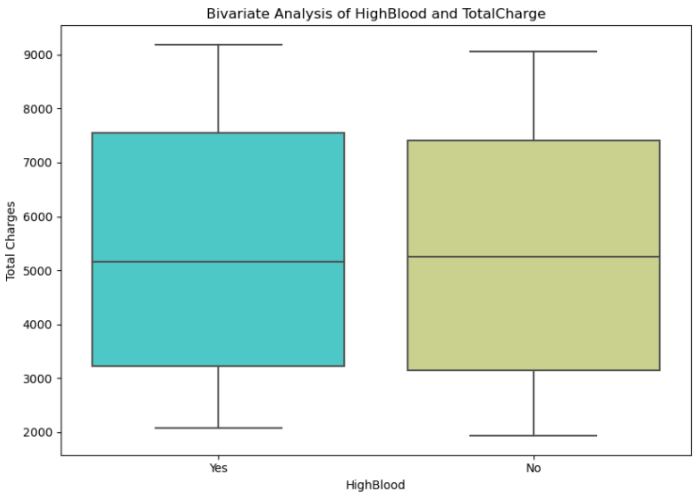
```
Initial_admin_counts = df["Initial_admin"].value_counts()
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Univariate Analysis of Initial_admin
ax1.pie(Initial_admin_counts, labels=Initial_admin_counts.index, autopct='%1.1f%%', startangle=140)
ax1.set_title('Univariate Analysis of Initial_admin')
ax1.axis('equal')

# Bivariate Analysis of Initial_admin and TotalCharge
sns.boxplot(x=df["Initial_admin"], y=df["TotalCharge"], ax=ax2)
ax2.set_title('Bivariate Analysis of Initial_admin and TotalCharge')
ax2.set_xlabel('Initial_admin')
ax2.set_ylabel('Total Charges')

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

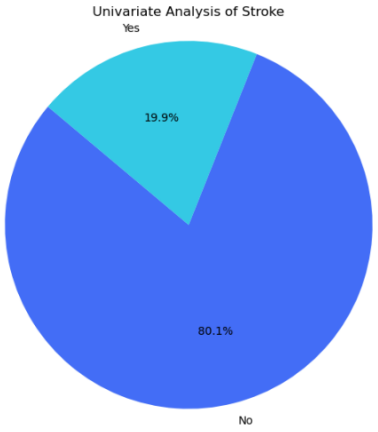


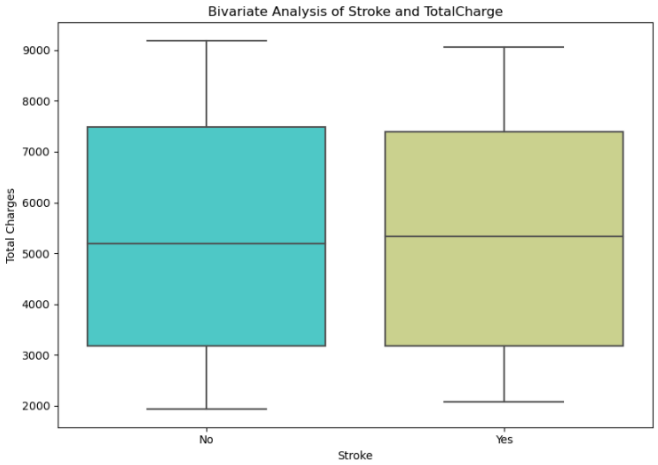

```
# Generate visualizations for other variables
variables = ["HighBlood", "Stroke", "Diabetes"]

for var in variables:
    counts = df[var].value_counts()
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

    # Pie Chart
    ax1.pie(counts, labels=counts.index, autopct='%1.1f%%', colors=sns.color_palette("rainbow"),
startangle=140)
    ax1.set_title(f'Univariate Analysis of {var}')
    ax1.axis('equal')

    # Box Plot
    sns.boxplot(x=df[var], y=df["TotalCharge"], ax=ax2, palette="rainbow")
    ax2.set_title(f'Bivariate Analysis of {var} and TotalCharge')
```

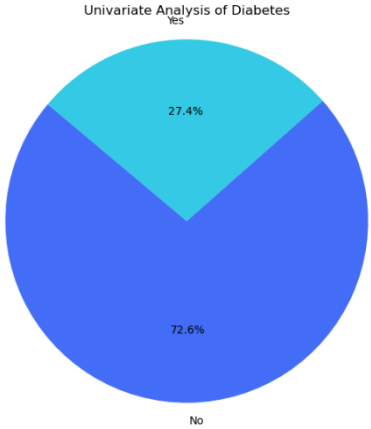
```
ax2.set_xlabel(var)
ax2.set_ylabel('Total Charges')

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

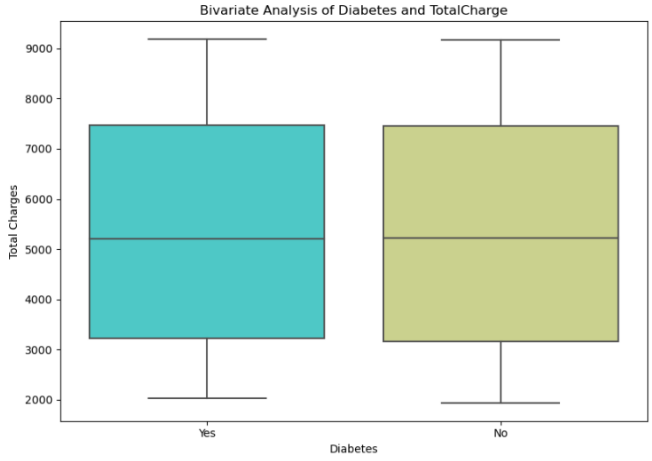












4. Describe your data transformation goals that align with your research question and the steps used to transform the data to achieve the goals, including the annotated code.

   The medical data set, in some variation, has been used for all the analysis done so far. Each iteration of the data set has been somewhat "cleaned" from the previous analysis done. The medical_clean data set did require some transformation to align with the predictive modeling needed.
   First the data needed to be inspected, to do that a display data types and visual inspection of the first file rows of data were called via coding.

```
# Display data types
df.info()

# Visually inspect df
pd.set_option("display.max_columns", None)
df.head(5)
```

   It was determined after inspecting the data that the proposed independent variables were of type 'object', 'int64', or 'float64'. Some of these data points needed to be changed.

   The variables that were of "object" type needed to be converted to "category" or "bool". The categorical column of "Initial_admin were adjusted with one hot encoding as explained by Dr. Middleton (Western Governors University, 2022) in the course videos.
   Finally once all the variables have been converted/adjusted a copy of the reduced data frame will be renamed to model_df.

```
# Data Transformation needed

# Update currency to 3 decimal places

df["Income"] = df["Income"].astype(int)



# Update Vitamin D levels to 3 decimal places

df["VitD_levels"] = df["VitD_levels"].astype(int)



# Update Initial days to 3 decimal places

df["Initial_days"] = df["Initial_days"].astype(int)
```

```
# Update Additional Charges to 3 decimal places

df["Additional_charges"] = df["Additional_charges"].astype(int)


# Convert columns to boolean

bool_mapping = {"Yes": 1, "No": 0}

columns_to_convert = ["HighBlood", "Stroke", "Diabetes"]

for col in columns_to_convert:

    df[col] = df[col].map(bool_mapping)




# Convert columns to category

df["Initial_admin"] = df["Initial_admin"].astype("category")


# Generate columns of dummy values

initial_admit_df = pd.get_dummies(data=df["Initial_admin"], drop_first=True)


# Create new df with model variables

model_df = df[[ "Children", "Age", "Income", "VitD_levels", "Doc_visits", "Full_meals_eaten",
"HighBlood", "Stroke", "Diabetes", "Initial_days", "Additional_charges"]].copy()




# Dummies for Initial Admit

model_df["initial_admit_elect"] = initial_admit_df["Emergency Admission"].astype(int)

model_df["initial_admit_obs"] = initial_admit_df["Observation Admission"].astype(int)
```

```
# Assuming "TotalCharge" is in df but not in model_df, copy it to model_df for analysis

model_df["TotalCharge"] = df["TotalCharge"]
```

```
# Visually inspect df

pd.set_option("display.max_columns", None)

model_df.head(5)
```

```
# Save model_df to a CSV file

model_df.to_csv("model_df.csv", index=False)

print("model_df has been saved to model_df.csv'.")
```

CaseOrder	Children	Age	Income	VitD_levels	Doc_visits	Full_meals_eaten	HighBlood	Stroke	Diabetes	Initial_days	Additional_charges	initial_admit_elect	initial_admit_obs	TotalCharge
1	1	53	86575	19	6	0	1	0	1	10	17939	1	0	3726.702860
2	3	51	46805	18	4	2	1	0	0	15	17612	1	0	4193.190458
3	3	53	14370	18	4	1	1	0	1	4	17505	0	0	2434.234222
4	0	78	39741	16	4	1	0	1	0	1	12993	0	0	2127.830423
5	1	22	1209	17	5	0	0	0	0	1	3716	0	0	2113.073274

```
model_df has been saved to model_df.csv'.
```

The prepared data set is saved as “D208_Austin_Tresa_1.ipynb and provided in the submission as well as a csv of the new dataset listed as model_df.csv.

Part IV: Model Comparison and Analysis

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

1. To construct an initial multiple linear regression model using all identified independent variables, Statsmodels (sm) was utilized. The code was developed with the assistance of resources from Data to Fish (2024), specifically the "Linear Regression in Python using Statsmodels" guide. The following code demonstrates a simple linear regression where y is the dependent variable and X represents the independent variables.

```
# Initial Multiple Linear Regression Model

y = model_df["TotalCharge"]
```

```
X = model_df[[

   "Children", "Age", "Income", "VitD_levels", "Doc_visits", "Full_meals_eaten", "HighBlood", "Stroke", "Diabetes", "Additional_charges" ,"initial_admit_elect", "initial_admit_obs"

]]

X = sm.add_constant(X)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())
```

OLS Regression Results

Dep. Variable:	TotalCharge	R-squared:	0.013
Model:	OLS	Adj. R-squared:	0.012
Method:	Least Squares	F-statistic:	11.21
Date:	Sat, 01 Jun 2024	Prob (F-statistic):	1.09e-22
Time:	16:51:41	Log-Likelihood:	-90995.
No. Observations:	10000	AIC:	1.820e+05
Df Residuals:	9987	BIC:	1.821e+05
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5106.1686	229.843	22.216	0.000	4655.629	5556.708
Children	22.9999	10.023	2.295	0.022	3.353	42.647
Age	1.7466	3.169	0.551	0.582	-4.465	7.958
Income	-0.0009	0.001	-1.197	0.231	-0.002	0.001
VitD_levels	-4.3458	10.637	-0.409	0.683	-25.196	16.504
Doc_visits	-11.5732	20.737	-0.558	0.577	-52.223	29.076
Full_meals_eaten	-33.7700	21.513	-1.570	0.117	-75.940	8.400
HighBlood	88.3388	122.615	0.720	0.471	-152.011	328.689
Stroke	-17.0504	54.471	-0.313	0.754	-123.825	89.724
Diabetes	58.3829	48.646	1.200	0.230	-36.973	153.739
Additional_charges	0.0002	0.013	0.015	0.988	-0.026	0.026
initial_admit_elect	447.8298	53.338	8.396	0.000	343.277	552.383
initial_admit_obs	-37.2896	61.719	-0.604	0.546	-158.272	83.693

Omnibus:	42757.564	Durbin-Watson:	0.179
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1233.509
Skew:	0.069	Prob(JB):	1.40e-268
Kurtosis:	1.285	Cond. No.	5.42e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.42e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

The results provide information that is useful as we venture down the analysis path. The R-squared value of 1.3% advises that this model only explains about of the variance in our independent variable “TotalCharge”. The large condition of 5.24e+05 indicates potential multicollinearity as well.
Additional work to remove non-significant columns will be done.

Also using a residual standard error to evaluate fit of the regression model and dataset can be beneficial here according to Bobbit(2021). This value measures the standard deviation of the

residuals in the regression model. It's calculated in python using the following code and the smaller the residual standard error is the better a regression model fits a dataset.

```
# Residual Standard Error
results.resid.std(ddof=X.shape[1])
```

With a residual value of 2167.1506719385116 there appears to be a difference between the observed and predicted values of TotalCharge for some data points in the model. After removal of data points, another check of the residual value will be done.

2. Justify a statistically based feature selection procedure or a model evaluation metric to reduce the initial model in a way that aligns with the research question "What factors can predict the total charges a patient will incur during their initial hospital stay?"

The best approach appears to be by starting with looking at multicollinearity using the Variance Inflation Factor (VIF) the below code helps identify the values for each variable. This information will show if there is any significant multicollinearity among the independent variables.

Then using backwards elimination using all dependent variables and removing one variable at a time based on its significance appears the be the best model to use with this data set after the initial values above (Grover, 2021).

3. Checking the Variance Inflation Factor (VIF), we see that # Calculate VIF

```
# Calculate VIF
def calculate_vif(X):
   vif = pd.DataFrame()
   vif["variables"] = X.columns
   vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
   return vif

X = model_df[[
    "Children", "Age", "Income", "VitD_levels", "Doc_visits", "Full_meals_eaten",
"HighBlood", "Stroke", "Diabetes",  "Additional_charges" ,"initial_admit_elect",
"initial_admit_obs"
]]

X = sm.add_constant(X)

vif_df = calculate_vif(X)
print(vif_df)
```

variables	VIF
const	112.482775
Children	1.001264
Age	9.105978
Income	1.001405
VitD_levels	1.002804

Doc_visits	1.001224
Full_meals_eaten	1.001413
HighBlood	7.737825
Stroke	1.008163
Diabetes	1.001854
Additional_charges	15.987277
initial_admit_elect	1.514159
initial_admit_obs	1.494496

The next path includes incorporating backwards elimination for each variable. By looking at the p values and removing the highest p value each time, we are looking to get down to variables with p values < 0.05 using this method.

Based on the above, Additonal_charges is removed from the df before continuing as it has a VIF over 10 which indicates multicollinearity.

After initial model is complete and it's determined that we need to remove variables one at a time to determine which ones are related to the dependent variable, a new model is created and scaled so we can effectively compare the data. The scaler fits the data and then transforms it and converts the columns back to the same names as the original data frame.

```
# Apply MinMaxScaler to df

scaler = MinMaxScaler()

regress_df = pd.DataFrame(scaler.fit_transform(model_df), columns=model_df.columns)

# Display the scaled DataFrame

print(regress_df)
```

```
     Children     Age   Income  VitD_levels  Doc_visits  Full_meals_eaten  \
0         0.1  0.492958  0.417301     0.588235       0.625          0.000000
1         0.3  0.464789  0.225264     0.529412       0.375          0.285714
2         0.3  0.492958  0.068645     0.529412       0.375          0.142857
3         0.0  0.845070  0.191154     0.411765       0.375          0.142857
4         0.1  0.056338  0.005094     0.470588       0.500          0.000000
...       ...       ...       ...          ...         ...               ...
9995      0.2  0.098592  0.221217     0.411765       0.375          0.285714
9996      0.4  0.971831  0.071605     0.529412       0.500          0.000000
9997      0.3  0.380282  0.317550     0.470588       0.375          0.285714
9998      0.3  0.352113  0.142678     0.588235       0.500          0.285714
9999      0.8  0.732394  0.301929     0.529412       0.500          0.000000

     HighBlood  Stroke  Diabetes  Initial_days  Additional_charges  \
0          1.0     0.0       1.0      0.128571            0.539849
1          1.0     0.0       0.0      0.200000            0.527933
2          1.0     0.0       1.0      0.042857            0.524033
3          0.0     1.0       0.0      0.000000            0.359608
4          0.0     0.0       0.0      0.000000            0.021537
```

```
...     ...   ...   ...   ...        ...
9995    1.0   0.0   0.0   0.714286   0.211435
9996    1.0   0.0   1.0   0.957143   0.924966
9997    1.0   0.0   0.0   0.985714   0.442987
9998    0.0   0.0   0.0   0.885714   0.169673
9999    0.0   0.0   0.0   0.985714   0.310411

     initial_admit_elect  initial_admit_obs  TotalCharge
0                    1.0                0.0     0.246933
1                    1.0                0.0     0.311343
2                    0.0                0.0     0.068475
3                    0.0                0.0     0.026168
4                    0.0                0.0     0.024130
...                  ...                ...          ...
9995                 1.0                0.0     0.678314
9996                 0.0                0.0     0.801304
9997                 0.0                0.0     0.875146
9998                 1.0                0.0     0.787882
9999                 0.0                1.0     0.821444
                [10000 rows x 14 columns]
```

```
# Model Reduction # 1: Find p-value above 0.05

y = regress_df.TotalCharge

X = regress_df [[ "Children", "Age", "Income", "VitD_levels", "Doc_visits", "Full_meals_eaten",
"HighBlood", "Stroke", "Diabetes", "initial_admit_elect", "initial_admit_obs"]].assign(const=1)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     12.24
Date:                Sat, 01 Jun 2024   Prob (F-statistic):           3.02e-23
Time:                        16:59:50   Log-Likelihood:                -90995.
No. Observations:               10000   AIC:                         1.820e+05
Df Residuals:                    9988   BIC:                         1.821e+05
Df Model:                          11
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Children             23.0028     10.021      2.296      0.022       3.360      42.645
Age                   1.7926      1.050      1.706      0.088      -0.267       3.852
Income               -0.0009      0.001     -1.197      0.231      -0.002       0.001
VitD_levels          -4.3479     10.635     -0.409      0.683     -25.195      16.499
Doc_visits          -11.5765     20.735     -0.558      0.577     -52.222      29.069
Full_meals_eaten    -33.7663     21.511     -1.570      0.117     -75.932       8.399
HighBlood            90.0978     44.088      2.044      0.041       3.676     176.520
Stroke              -16.9770     54.259     -0.313      0.754    -123.335      89.381
Diabetes             58.3937     48.638      1.201      0.230     -36.947     153.735
initial_admit_elect 447.9233     52.987      8.454      0.000     344.059     551.788
initial_admit_obs   -37.3118     61.699     -0.605      0.545    -158.255      83.632
const              5105.6069    226.910     22.501      0.000    4660.817    5550.397
==============================================================================
Omnibus:                    42757.974   Durbin-Watson:                   0.179
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1233.496
Skew:                           0.069   Prob(JB):                    1.41e-268
Kurtosis:                       1.285   Cond. No.                     5.20e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.2e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model Reduction # 2: Find p-value above 0.05 (removed Stroke with p-value of .754)

y = regress_df.TotalCharge

X = regress_df [["Children", "Age", "Income", "VitD_levels", "Doc_visits", "Full_meals_eaten", "HighBlood", "Diabetes", "initial_admit_elect", "initial_admit_obs"]].assign(const=1)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     13.45
Date:                Sat, 01 Jun 2024   Prob (F-statistic):           8.35e-24
Time:                        17:00:28   Log-Likelihood:                -90995.
No. Observations:               10000   AIC:                         1.820e+05
Df Residuals:                    9989   BIC:                         1.821e+05
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Children       22.9876     10.020      2.294      0.022       3.346      42.629
Age             1.7887      1.050      1.703      0.089      -0.270       3.848
Income         -0.0009      0.001     -1.198      0.231      -0.002       0.001
VitD_levels    -4.3723     10.635     -0.411      0.681     -25.218      16.474
Doc_visits    -11.5603     20.734     -0.558      0.577     -52.204      29.083
Full_meals_eaten -33.7824     21.510     -1.571      0.116     -75.946       8.381
HighBlood      89.9951     44.085      2.041      0.041       3.580     176.411
Diabetes       58.3054     48.635      1.199      0.231     -37.030     153.641
initial_admit_elect  448.0562     52.983      8.457      0.000     344.200     551.913
initial_admit_obs    -37.3133     61.697     -0.605      0.545    -158.251      83.625
const         5102.8448    226.728     22.506      0.000    4658.412    5547.278
==============================================================================
Omnibus:                    42759.746   Durbin-Watson:                   0.179
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1233.470
Skew:                           0.069   Prob(JB):                    1.43e-268
Kurtosis:                       1.285   Cond. No.                     5.20e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.2e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model Reduction # 3: Find p-value above 0.05 (removed VitD_levels with p-value of .681)

y = regress_df.TotalCharge

X = regress_df [["Children", "Age", "Income", "Doc_visits", "Full_meals_eaten", "HighBlood", "Diabetes", "initial_admit_elect", "initial_admit_obs"]].assign(const=1)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     14.93
Date:                Sat, 01 Jun 2024   Prob (F-statistic):           2.26e-24
Time:                        17:01:19   Log-Likelihood:                -90995.
No. Observations:               10000   AIC:                         1.820e+05
Df Residuals:                    9990   BIC:                         1.821e+05
Df Model:                           9
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
Children               22.9454     10.019      2.290      0.022       3.306      42.585
Age                     1.7849      1.050      1.699      0.089      -0.274       3.844
Income                 -0.0009      0.001     -1.193      0.233      -0.002       0.001
Doc_visits            -11.6550     20.732     -0.562      0.574     -52.294      28.984
Full_meals_eaten      -33.9997     21.502     -1.581      0.114     -76.149       8.149
HighBlood              89.9083     44.083      2.040      0.041       3.497     176.319
Diabetes               58.7897     48.619      1.209      0.227     -36.514     154.093
initial_admit_elect   447.5185     52.964      8.449      0.000     343.698     551.339
initial_admit_obs     -37.3398     61.694     -0.605      0.545    -158.273      83.593
const                5027.5036    133.506     37.657      0.000    4765.804    5289.203
==============================================================================
Omnibus:                    42757.411   Durbin-Watson:                   0.179
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1233.540
Skew:                           0.069   Prob(JB):                    1.38e-268
Kurtosis:                       1.285   Cond. No.                     3.12e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.12e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model Reduction #4: Find p-value above 0.05 (removed Doc_visits with p-value of .574)

y = regress_df.TotalCharge

X = regress_df [["Children", "Age", "Income", "Full_meals_eaten", "HighBlood", "Diabetes", "initial_admit_elect", "initial_admit_obs"]].assign(const=1)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     16.76
Date:                Sat, 01 Jun 2024   Prob (F-statistic):           6.17e-25
Time:                        17:02:11   Log-Likelihood:                -90995.
No. Observations:               10000   AIC:                         1.820e+05
Df Residuals:                    9991   BIC:                         1.821e+05
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Children              22.9610     10.019      2.292      0.022       3.322      42.600
Age                    1.7806      1.050      1.695      0.090      -0.278       3.839
Income                -0.0009      0.001     -1.201      0.230      -0.002       0.001
Full_meals_eaten     -33.9614     21.502     -1.579      0.114     -76.109       8.186
HighBlood             89.6869     44.079      2.035      0.042       3.282     176.091
Diabetes              58.4294     48.613      1.202      0.229     -36.862     153.721
initial_admit_elect  447.0456     52.956      8.442      0.000     343.242     550.850
initial_admit_obs    -38.0922     61.677     -0.618      0.537    -158.992      82.808
const               4970.0923     85.989     57.799      0.000    4801.537    5138.648
==============================================================================
Omnibus:                    42747.238   Durbin-Watson:                   0.179
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1233.868
Skew:                           0.069   Prob(JB):                    1.17e-268
Kurtosis:                       1.285   Cond. No.                     2.21e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.21e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model Reduction #5: Find p-value above 0.05 (removed initial_admit_obs with p-value of .537)

```
y = regress_df.TotalCharge

X = regress_df [[ "Children", "Age", "Income", "Full_meals_eaten", "HighBlood", "Diabetes",
      "initial_admit_elect"]].assign(const=1)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.013
Method:                 Least Squares   F-statistic:                     19.10
Date:                Sat, 01 Jun 2024   Prob (F-statistic):           1.62e-25
Time:                        17:03:02   Log-Likelihood:                -90995.
No. Observations:               10000   AIC:                         1.820e+05
Df Residuals:                    9992   BIC:                         1.821e+05
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Children      22.9692     10.018      2.293      0.022       3.331      42.607
Age            1.7892      1.050      1.704      0.088      -0.269       3.848
Income        -0.0009      0.001     -1.208      0.227      -0.002       0.001
Full_meals_eaten   -34.0966     21.500     -1.586      0.113     -76.240       8.047
HighBlood     89.5165     44.077      2.031      0.042       3.116     175.917
Diabetes      58.5799     48.611      1.205      0.228     -36.708     153.867
initial_admit_elect   465.8267     43.353     10.745      0.000     380.847     550.806
const       4951.2113     80.368     61.606      0.000    4793.673    5108.750
==============================================================================
Omnibus:                    42742.705   Durbin-Watson:                   0.179
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1234.007
Skew:                           0.069   Prob(JB):                     1.09e-268
Kurtosis:                       1.285   Cond. No.                     1.90e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.9e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model Reduction #6: Find p-value above 0.05 (removed Diabetes with p-value of .228)

```
y = regress_df.TotalCharge

X = regress_df[[ "Children", "Age", "Income", "Full_meals_eaten", "HighBlood",
      "initial_admit_elect"]].assign(const=1)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     22.03
Date:                Sat, 01 Jun 2024   Prob (F-statistic):           6.62e-26
Time:                        17:03:42   Log-Likelihood:                -90996.
No. Observations:               10000   AIC:                         1.820e+05
Df Residuals:                    9993   BIC:                         1.821e+05
Df Model:                           6
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
Children               23.2418     10.016      2.320      0.020       3.608      42.875
Age                     1.7933      1.050      1.708      0.088      -0.265       3.852
Income                 -0.0009      0.001     -1.220      0.222      -0.002       0.001
Full_meals_eaten      -33.8500     21.499     -1.574      0.115     -75.993       8.293
HighBlood              89.1928     44.077      2.024      0.043       2.792     175.593
initial_admit_elect   465.3695     43.352     10.735      0.000     380.391     550.348
const                4966.9573     79.301     62.634      0.000    4811.512    5122.403
==============================================================================
Omnibus:                    42777.976   Durbin-Watson:                   0.179
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1232.886
Skew:                           0.069   Prob(JB):                    1.91e-268
Kurtosis:                       1.285   Cond. No.                     1.86e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.86e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model Reduction #7: Find p-value above 0.05 (removed Income with p-value of .222)

y = regress_df.TotalCharge

X = regress_df[["Children", "Age", "Full_meals_eaten", "HighBlood", "initial_admit_elect"]].assign(const=1)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     26.14
Date:                Sat, 01 Jun 2024   Prob (F-statistic):           2.52e-26
Time:                        17:04:15   Log-Likelihood:                -90997.
No. Observations:               10000   AIC:                         1.820e+05
Df Residuals:                    9994   BIC:                         1.820e+05
Df Model:                           5
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
Children               23.1496     10.016      2.311      0.021       3.516      42.783
Age                     1.8091      1.050      1.723      0.085      -0.249       3.867
Full_meals_eaten      -33.5579     21.498     -1.561      0.119     -75.699       8.583
HighBlood              89.2545     44.078      2.025      0.043       2.852     175.657
initial_admit_elect   466.5237     43.343     10.764      0.000     381.563     551.484
const                4927.8538     72.539     67.934      0.000    4785.662    5070.045
==============================================================================
Omnibus:                    42767.982   Durbin-Watson:                   0.179
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1233.234
Skew:                           0.069   Prob(JB):                    1.61e-268
Kurtosis:                       1.285   Cond. No.                         200.
==============================================================================

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

Model Reduction #8: Find p-value above 0.05 (removed Full_meals_eaten with p-value of .119)

```
y = regress_df.TotalCharge

X = regress_df[[ "Children", "Age", "HighBlood", "initial_admit_elect"]].assign(const=1)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     32.06
Date:                Sat, 01 Jun 2024   Prob (F-statistic):           1.36e-26
Time:                        17:04:48   Log-Likelihood:                -90998.
No. Observations:               10000   AIC:                         1.820e+05
Df Residuals:                    9995   BIC:                         1.820e+05
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
Children                      23.0922     10.017      2.305      0.021       3.457      42.727
Age                            1.7953      1.050      1.710      0.087      -0.263       3.854
HighBlood                     88.2412     44.077      2.002      0.045       1.842     174.641
initial_admit_elect          466.0934     43.345     10.753      0.000     381.129     551.058
const                       4895.7418     69.566     70.376      0.000    4759.379    5032.105
==============================================================================
Omnibus:                    42731.089   Durbin-Watson:                   0.178
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1234.420
Skew:                           0.069   Prob(JB):                     8.89e-269
Kurtosis:                       1.284   Cond. No.                         193.
==============================================================================

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

Model Reduction #9: Find p-value above 0.05 (removed Age with p-value of .087)

```
y = regress_df.TotalCharge

X = regress_df[[ "Children", "HighBlood", "initial_admit_elect"]].assign(const=1)

model = sm.OLS(y, X)

results = model.fit()

print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.012
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     41.77
Date:                Sat, 01 Jun 2024   Prob (F-statistic):           8.12e-27
Time:                        17:05:23   Log-Likelihood:                -90999.
No. Observations:               10000   AIC:                         1.820e+05
Df Residuals:                    9996   BIC:                         1.820e+05
Df Model:                           3
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
Children                    23.2612     10.017      2.322      0.020       3.625      42.897
HighBlood                   88.7779     44.080      2.014      0.044       2.372     175.183
initial_admit_elect        465.7512     43.349     10.744      0.000     380.779     550.723
const                     4991.4092     41.333    120.762      0.000    4910.389    5072.430
==============================================================================
Omnibus:                    42677.922   Durbin-Watson:                   0.178
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1236.091
Skew:                           0.069   Prob(JB):                     3.86e-269
Kurtosis:                       1.283   Cond. No.                         7.99
==============================================================================

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

Residual Standard Error

results.resid.std(ddof=X.shape[1])
0.29928001854184033

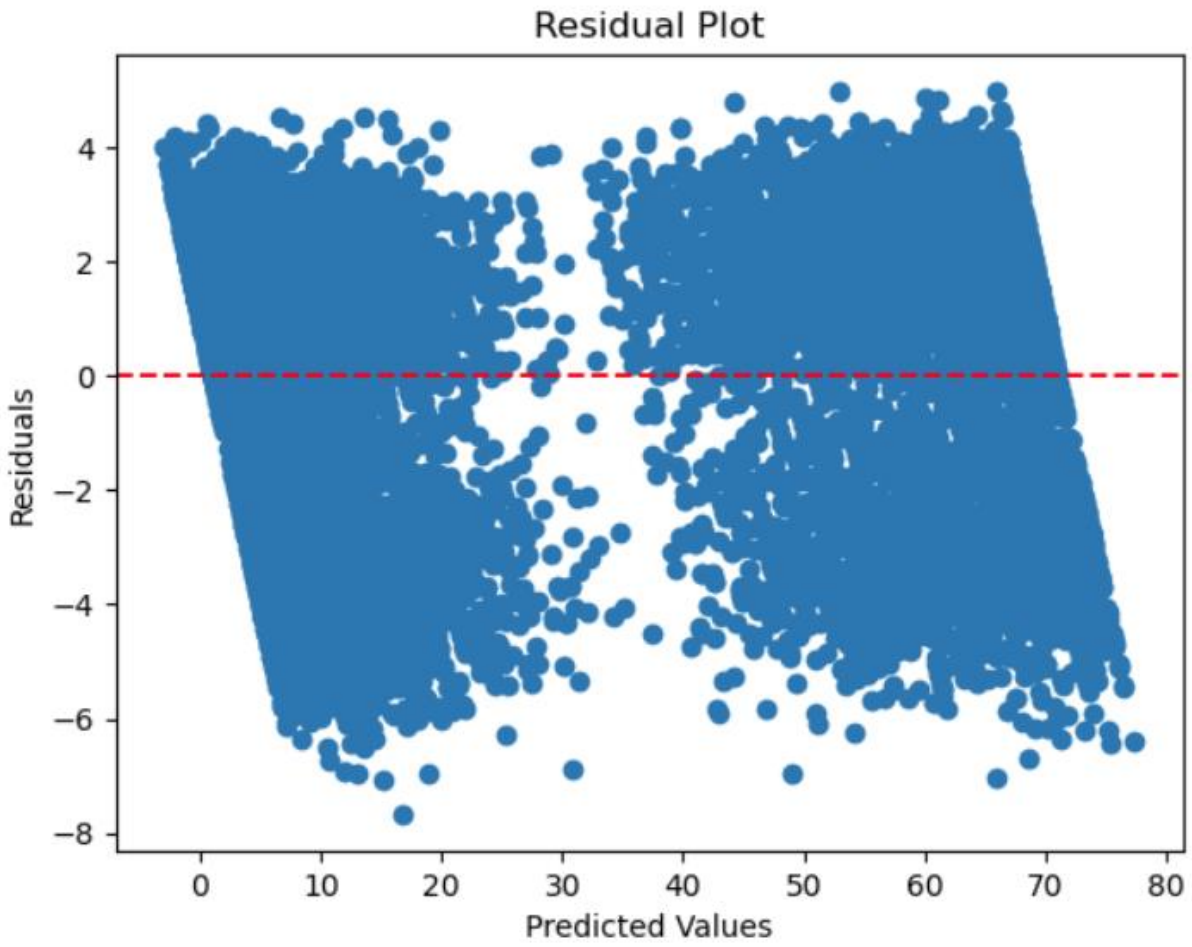
E. Analyze the data set using your reduced linear regression model by doing the following:

1. The initial multiple linear regression model included 14 variables that appeared to relate to how long a patient spent in the hospital. The R-squared value of .0002 advises that this model only explains about .02% of the variance in the data. Using backwards elimination, we have ruled out all variables that were thought to have significance and now we have an R-squared of 0, which means none of the variables are significant.

2. All of the output and calculations for the reduced linear regression models are listed above in Part IV, section D3.
For those calculations, the following residual plot was created.

```
# Residual Plot
y = model_df["TotalCharge"]
X = model_df[["Children", "Age", "Income", "VitD_levels", "Doc_visits", "Full_meals_eaten",
"HighBlood", "Stroke", "Diabetes", "Initial_days", "Additional_charges" , “initial_admit_elect
", "initial_admit_obs"]]

data = pd.concat([X, y], axis=1).dropna()

X = data.drop(columns=["Initial_days"])
y = data["Initial_days"]

X = X.apply(pd.to_numeric, errors='coerce')
y = pd.to_numeric(y, errors='coerce')

X = sm.add_constant(X)
```

```
model = sm.OLS(y, X)
results = model.fit()

predicted_values = results.fittedvalues
residuals = results.resid

plt.scatter(predicted_values, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.show()
```



Based on the visualization above, there appear to be issues with the model. Heteroscedasticity could be occurring based on how the spread of the residuals increases with the predicted values. Also there appears to be a patter in the graph which could also indicate that the relationship is non-linear(Bobbit, 2021).
There is also a decided funnel shape which could signal non-linearity in the model meaning the relationship between the variables is not aligned.

Also as listed above, the residual standard error for the initial model is found by using

```
results.resid.std(ddof=X.shape[1])
```

And the value equates to 26.301939165271886

The residual standard error of the adjusted (regress) model is found with the same code and it equate to 0.29928001854184033. This small number indicates this model may be a good fit.

3. An executable error-free copy of the code used to support the implementation of the linear regression models is attached and listed as D208_Austin_Tresa_2.ipynb.

Part V: Data Summary and Implications

F. Summarize your findings and assumptions

The following is the regression equation using the values from our analysis:

Y = 0.4216 +0.0321(Children) +0.0123(HighBlood)+0.0643(Initial_admin_elect)

The coefficients are interpreted as follows:

1. Children (0.0321) - for each additional child the value of Total Charge is expected to increase by approximately $0.03 with all other variables being constant.
2. HighBlood(0.0123) – if a patient has high blood pressure their Total Charge is expected to increase by $0.01 with all other variables being constant.
3. Initial_admin_elect(0.0643) – if a patient is admitted for an elective procedure their Total Charge is expected to increase by $0.06 with all other variables being constant.

The practical significance of the reduced model shows us the real-world relevance of the relationships in the model. While the statistical significance shows us that while an effect can exist, that effect size may not be large enough to matter in a practical application.
Our findings showed that number of children in the patient's household, whether or not the patient has high blood pressure, and elective admissions were statistically significant but the actual values indicate the effect of those on the patients' total charges is limited in impact.

Overall the reduced model explains a very small amount of the variance in TotalCharge. Even though many variables from the set were used, there could be others that may have a larger effect on the outcome. There may also be variables that aren't captured that play a larger role in the significance of the patient's total charges such as type of insurance they have or if a surgery is part of their stay.

2. Recommend a course of action based on your results.

The recommended course of action would be to expand the variables to capture more data points to see if there are other factors that correlate to the total charge a patient receives while staying in the hospital. Another recommended course of action would be to expand the sample size to more than 10,000 to see if that gives different results.

Part VI: Demonstration

G. Panopto video link

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=d251ac38-fecd-4200-b7e6-b1840174d563

H. Web Sources

Brownlee, J. (2020, August 28). RFE Feature Selection in Python. Machine Learning Mastery. Retrieved May 24, 2024, *from* https://machinelearningmastery.com/rfe-feature-selection-in-python/

Bobbitt, Z. (2021, May 11). How to interpret residual standard error. *Statology*. https://www.statology.org/how-to-interpret-residual-standard-error/

Data to Fish. (2024). Example of Multiple Linear Regression in Python. Retrieved May 24, 2024, *from* https://datatofish.com/multiple-linear-regression-python/

Grover, J. (2021). Short Python code for backward elimination with detailed explanation. *Medium*. Retrieved from https://groverjatin.medium.com/short-python-code-for-backward-elimination-with-detailed-explanation-52894a9a7880

Western Governors University. (2022, November). D208 - Webinar: Getting started with D208 Part I [Video]. Panopto. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=fa5c5de0-e9af-47c9-931d-b07d01014776&query=d208

I. Sources

Beyond Verse. (2023, January 23). The role of Python in big data and analytics. Medium. https://medium.com/@beyond_verse/the-role-of-python-in-big-data-and-analytics-2da818c4cbf

Bobbitt, Z. (2021, November 16). The five assumptions of multiple linear regression. Statology. https://www.statology.org/multiple-linear-regression-assumptions/

Boston University School of Public Health. (2013, January 17). Multiple linear regression analysis. In *BS704: Multivariable methods*. Boston University. https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_multivariable/bs704_multivariable7.html

NCRM. (2011, July 20). Overview. Using statistical regression methods in education research. ReStore. https://www.restore.ac.uk/srme/www/fac/soc/wie/research-new/srme/modules/mod3/1/index.html

Sharma, A. (2019) *Histograms with Matplotlib*. DataCamp. Retrieved May 24, 2024, from https://www.datacamp.com/tutorial/histograms-matplotlib

Statistics By Jim. (2021). Box Plot Explained with Examples. Retrieved May 24, 2024, from https://www.statisticsbyjim.com

Waqar, H. (2024). What is the get_dummies function in pandas? *Educative.io*. https://www.educative.io/answers/what-is-the-getdummies-function-in-pandas

Western Governors University. (n.d.). R or Python? Which is better for data analysis? Retrieved May 17, 2024, from https://www.wgu.edu/online-it-degrees/programming-languages/r-or-python.html