D208 – Predictive Modeling - Task 2

WGU M.S. Data Analytics

Lyssa Kline

October 3, 2024

A, Research Question

A1, Question

The research question in scope for this analysis is, "What patient characteristics or factors influence hospital readmissions?" Understanding the factors influencing patient readmissions can help medical organizations manage and potentially reduce costs. This insight could lead to more efficient allocation of resources, better staffing, and greater patient transparency. Analyzing the factors affecting readmissions performed could help identify gaps in patient care, which could lead to targeted efforts to provide actionable insights promptly and improve the quality of care to patients.

A2, Analysis Goals

The goal of the analysis is to determine which variables signify the need for patient readmissions. Analyzing the magnitude of each factor will help measure the impact of hospital readmissions. The primary goal of this analysis will be to provide actionable insights that can be used to ensure appropriate care is being performed during an initial visit, as well as improve the quality of care and provide actionable services promptly.

B, Method Justification

B1, Logistic Regression Model Assumptions

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

- The first assumption is the dependent must be binary or categorical depending on the type of regression model.
- The second assumption is that a logistic regression model assumes there are no highly influential outliers in the data. Outliers, especially in the independent variables, can disproportionately influence the model's coefficients, leading to misleading results.
- The third assumption is that the observations should be independent of each other, meaning that there should be no correlation or dependence between the observations.
- The last assumption would be that logistic regression assumes that there is no perfect multicollinearity among the independent variables. High multicollinearity can distort the estimated coefficients and reduce the interpretability of the model.

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 can create a strong ecosystem for data handling and visualizations, allowing ease of

implementation and flexibility to the model. 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
from statsmodels.graphics.mosaicplot import mosaic
import statsmodels.api as sm
from sklearn.feature_selection import RFE
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from category_encoders import BinaryEncoder
from imblearn.over_sampling import SMOTE
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score, log_loss
from sklearn.decomposition import PCA
import seaborn as sns
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 patient readmissions (dependent variable) and various potential influencing factors (independent variables). The dependent variable in the scope of 'ReAdmis' is a categorical field containing whether a patient has been readmitted to the hospital or not. For this analysis, logistic regression is the appropriate technique due to a variety of reasons.

Logistic regression is specifically designed to model relationships when the dependent variable is categorical. Logistic regression enables probabilistic prediction and classification, helping to assess the likelihood of patients being readmitted based on their characteristics and services performed. Furthermore, the model provides interpretable insights into how different factors influence these decisions, making it valuable in a healthcare setting for resource allocation and clinical decision-making.

Since the dependent variable (ReAdmis) is a categorical field with just two categories, logistic regression will allow us to code these as a binary 1 or 0 and allow us to predict the probability of the outcome, either Yes or No.

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 logistic 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 patient characteristics or factors influence hospital readmissions?" The dependent variable would be 'ReAdmis,' and the independent variables selected include key patient characteristics/factor information that may influence the need to be readmitted. To ensure adequate coverage of variables, there were 17 independent variables chosen for further analysis.

Dependent Variable chosen for analysis:

- ReAdmis – Data Type: Category, Categorical Variable – Describes if the patient was re-admitted or not. Based on summary statistics, there are 10000 data points containing two unique values: Yes or No.

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

```
       ReAdmis
count    10000
unique       2
top         No
freq      6331
```

Independent Variables were chosen for analysis:

- Age – Data Type: integer, Continuous Numerical Variable – The patient's age. Based on the summary statistics, there are 10000 data points containing a mean age of 53, with min being 18 and max being 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
```

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

- Doc_visits - Data Type: integer, Continuous Numerical Variable – Describes how many doctor visits a patient has had. Based on summary statistics, the variable has 10000 data points with a mean of 5, min being 1 and max being 9.

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

```
         Doc_visits
count  10000.000000
mean       5.012200
std        1.045734
min        1.000000
25%        4.000000
50%        5.000000
75%        6.000000
max        9.000000
```

- Soft_drink - Data Type: category, Categorical Variable – Describes if the patient drinks soft drinks or not. Based on summary statistics, the variable has 10000 data points with two unique yes or no fields, the top selection being no.

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

```
       Soft_drink
count       10000
unique          2
top            No
freq         7425
```

- 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 three 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
```

- 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 two 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 two 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
```

- 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 three 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
```

- Overweight - Data Type: Category, Categorical Variable – Whether the patient is overweight or not. Based on summary statistics, the variable has 10000 data points containing two 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 two 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 two 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 two 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
```

- Allergic_rhinitis - Data Type: Category, Categorical Variable – Whether the patient has allergic rhinitis or not. Based on summary statistics, the variable has 10000 data points containing two unique values, with No being the top one.

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

```
       Allergic_rhinitis
count              10000
unique                 2
top                   No
freq                6059
```

- 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 two 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
```

- Asthma - Data Type: Category, Categorical Variable – Whether the patient has asthma or not. Based on summary statistics, the variable has 10000 data points containing two unique values, with No being the top one.

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

```
        Asthma
count    10000
unique       2
top         No
freq      7107
```

- Services - Data Type: Category, Categorical Variable – Primary services the patient received. Based on summary statistics, the variable has 10000 data points containing four 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
```

- Initital_days - Data Type: numerical, Continuous Variable – Describes how many days the patient has been admitted to the hospital. Based on summary statistics, the variable has 10000 data points containing a mean of 34, min of 1 and max of 71.

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

```
        Initial_days
count   10000.000000
mean       34.455299
std        26.309341
min         1.001981
25%         7.896215
50%        35.836244
75%        61.161020
max        71.981490
```

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

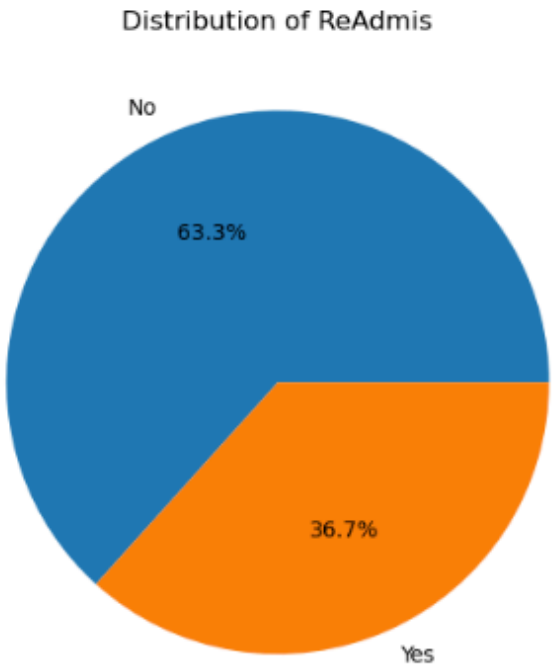
tendency, dispersion, shape, and outliers for a single variable. In comparison, bivariate statistics involve the analysis of two variables to explore the relationship between them. Bivariate analysis can be used to determine whether and how two variables are related to each other.

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 ReAdmis yes or no categories, as you can see, 'No' was performed the most, indicating that most patients were not readmitted. However, 36.7% of patients being readmitted seems high; we will continue to examine the factors influencing this in the below analysis.

```
# First examining the dependent variable for distribution of the Services performed on patients.
# Pie Chart Analysis on Services.
axes = df['ReAdmis'].value_counts().plot.pie(autopct='%1.1f%%')
axes.set_title('Distribution of ReAdmis')
axes.set_ylabel("")

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



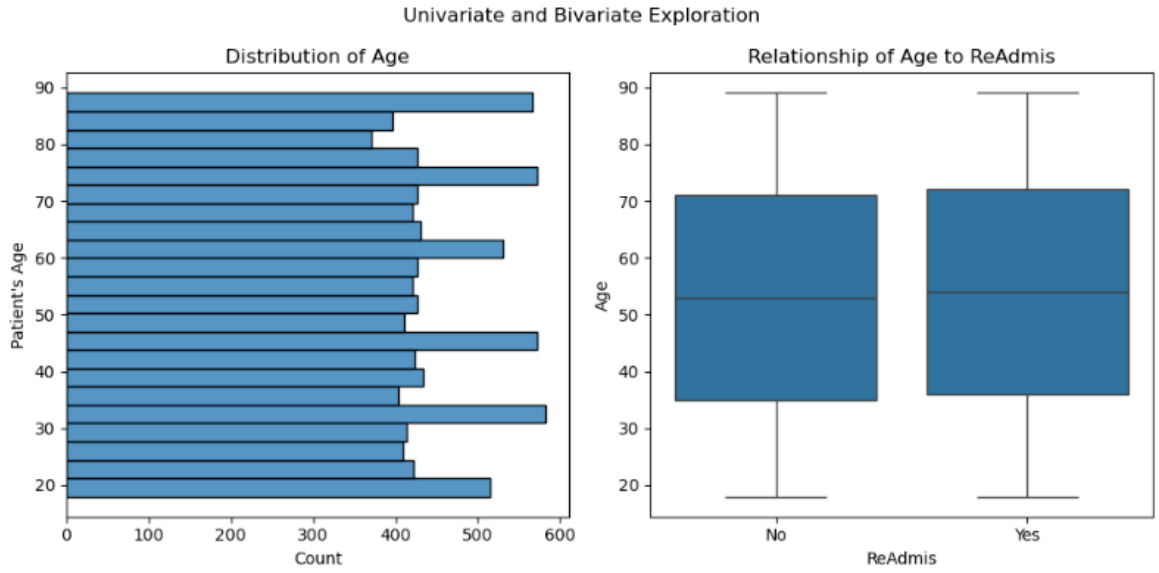
The chart below shows the distribution of the independent variable 'Age' using a Univariate analysis, as well as the distribution and correlation between the independent variable 'Age' and the dependent variable 'ReAdmis' using a Bivariate Analysis. The visualizations below show the categories contained in ReAdmis with the age dispersion inside each one. This helps to visualize the various age groups that were readmitted. The median age is very similar in both readmission categories, and age does not show a direct correlation to hospital readmissions.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Age  Dependent Variable: ReAdmis
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.
# Histogram Chart Analysis on Age.
sns.histplot(data=df, y='Age',ax=axes[0])
axes[0].set_title("Distribution of Age")
axes[0].set_ylabel("Patient's Age")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Age (continuous)
sns.boxplot(x="ReAdmis", y="Age", data=df, ax=axes[1])
axes[1].set_title("Relationship of Age to ReAdmis")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Age")

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



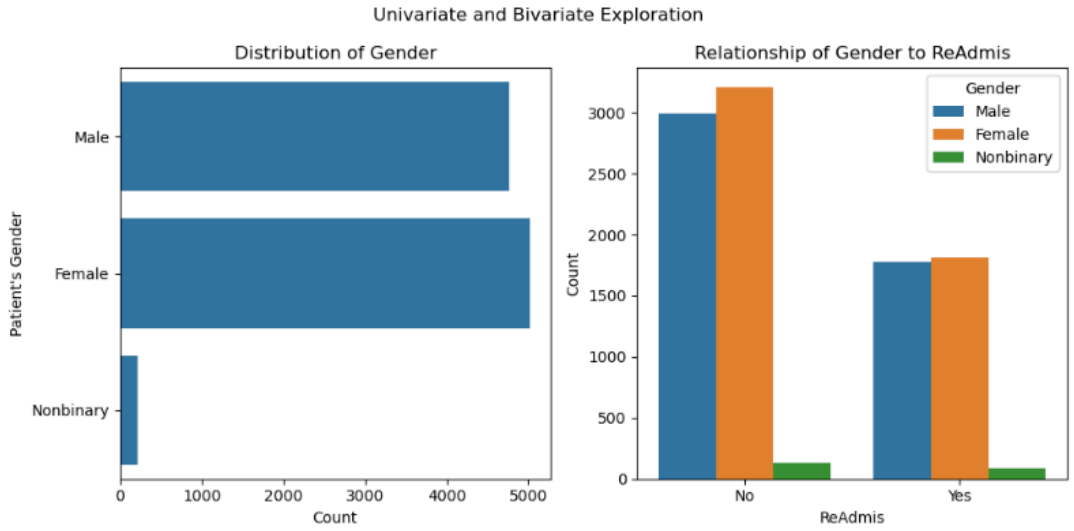
The chart below shows the distribution of the independent variable ‘Gender’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Gender’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show the distribution of the Gender category and then the way that it is broken out into the readmission categories. You can see there are more females than males, and they have similar rates of readmission as males. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Gender Dependent Variable: ReAdmis
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 Gender.
sns.countplot(data=df, y='Gender',ax=axes[0])
axes[0].set_title("Distribution of Gender")
axes[0].set_ylabel("Patient's Gender")
axes[0].set_xlabel("Count")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Gender (categorical)
sns.countplot(x="ReAdmis", hue="Gender", data=df, ax=axes[1])
axes[1].set_title("Relationship of Gender to ReAdmis")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")
axes[1].legend(title="Gender")

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



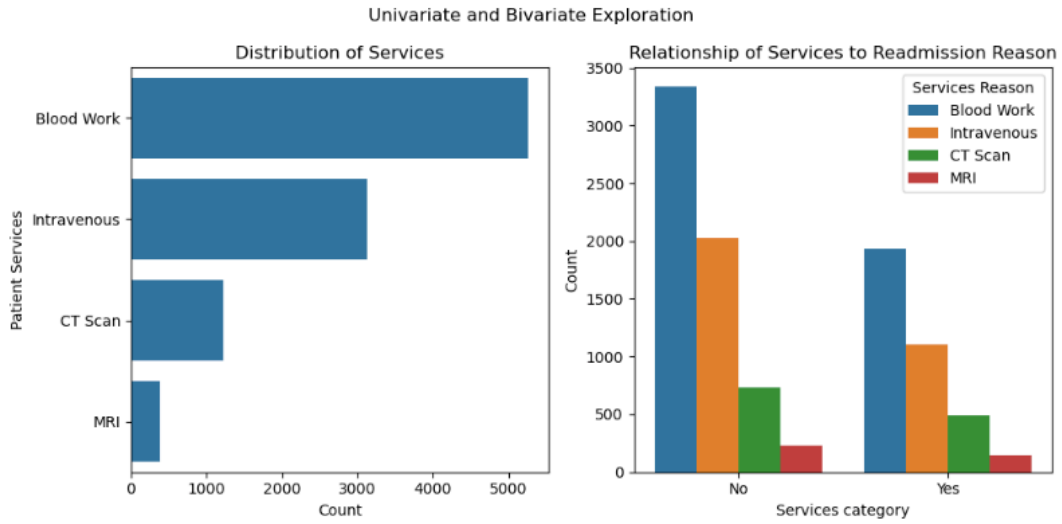
The chart below shows the distribution of the independent variable ‘Services’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Services’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show the dispersion of whether patients were re-admitted or not and the dispersion of the services performed within those categories. You can see how bloodwork is the most common service. Additionally, you can see the dispersion between admission and the services performed. There’s no obvious category that tends to be higher when the other is done, indicating no direct relationship between the fields. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Services Dependent Variable: ReAdmis
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 Services.
sns.countplot(data=df, y='Services',ax=axes[0])
axes[0].set_title("Distribution of Services")
axes[0].set_ylabel("Patient Services")
axes[0].set_xlabel("Count")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & services (categorical)
sns.countplot(x="ReAdmis", hue="Services", data=df, ax=axes[1])
axes[1].set_title("Relationship of Services to Readmission Reason")
axes[1].set_xlabel("Services category")
axes[1].set_ylabel("Count")
axes[1].legend(title="Services Reason")

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



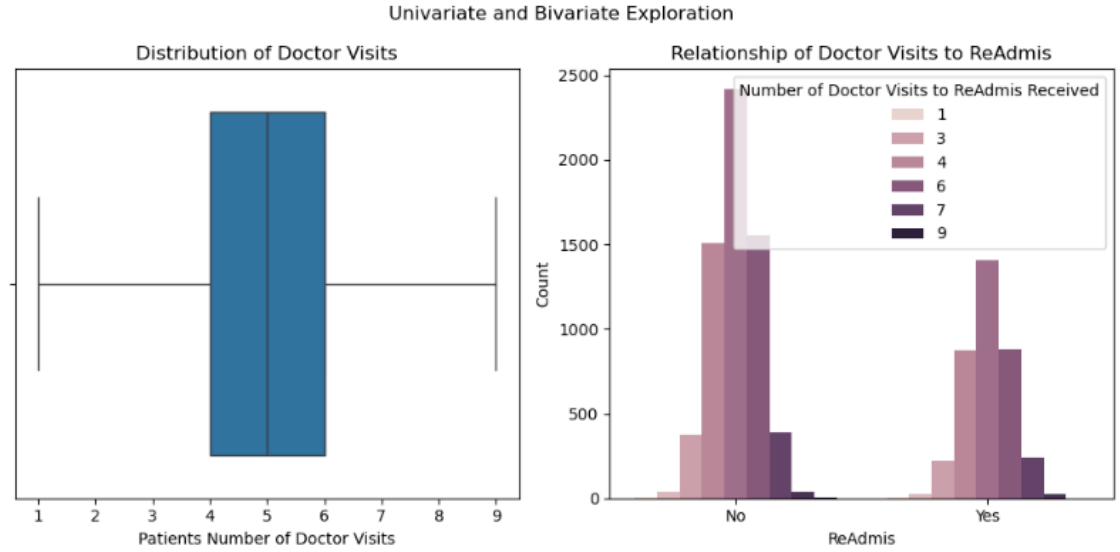
The chart below shows the distribution of the independent variable ‘Doc_visits’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Doc_visits’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show the dispersion of the number of doctor visits and then the dispersion of the readmission categories within that. The dispersion of doctor visits between readmission and no readmission is very similar between the two categories. Indicating doctor visits may not influence readmission. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Doc_visits Dependent Variable: ReAdmis
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.
# Box Plot Analysis on Doc_visits.
sns.boxplot(x='Doc_visits', data=df, ax=axes[0])
axes[0].set_title('Distribution of Doctor Visits')
axes[0].set_xlabel("Patients Number of Doctor Visits")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Doc_visits (continuous)
sns.countplot(x="ReAdmis", hue="Doc_visits", data=df, ax=axes[1])
axes[1].set_title("Relationship of Doctor Visits to ReAdmis")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")
axes[1].legend(title="Number of Doctor Visits to ReAdmis Received")

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



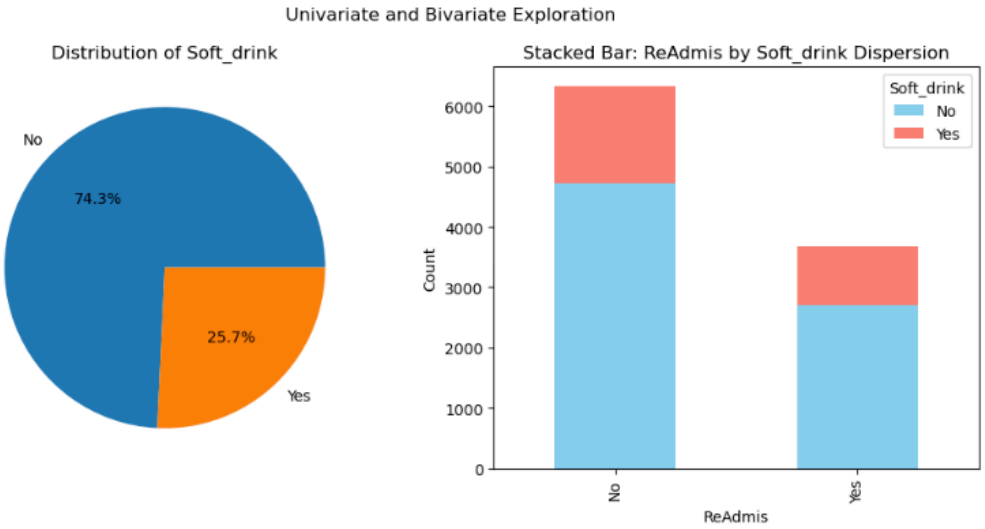
The chart below shows the distribution of the independent variable ‘Soft_drink’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Soft_drink’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show that more people sit in the no soft drink category, and the dispersion across the readmission categories makes sense for the number of patients. Indicating there is no obvious relationship between the fields. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Soft_drink Dependent Variable: ReAdmis
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 Soft_drink
df['Soft_drink'].value_counts().plot.pie(autopct='%1.1f%%', ax=axes[0])
axes[0].set_title('Distribution of Soft_drink')
axes[0].set_ylabel("")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Soft_drink (categorical)
crosstab_data = pd.crosstab(df['ReAdmis'], df['Soft_drink'])
crosstab_data.plot(kind='bar', stacked=True, ax=axes[1], color=['skyblue', 'salmon'])
axes[1].set_title("Stacked Bar: ReAdmis by Soft_drink Dispersion")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")

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



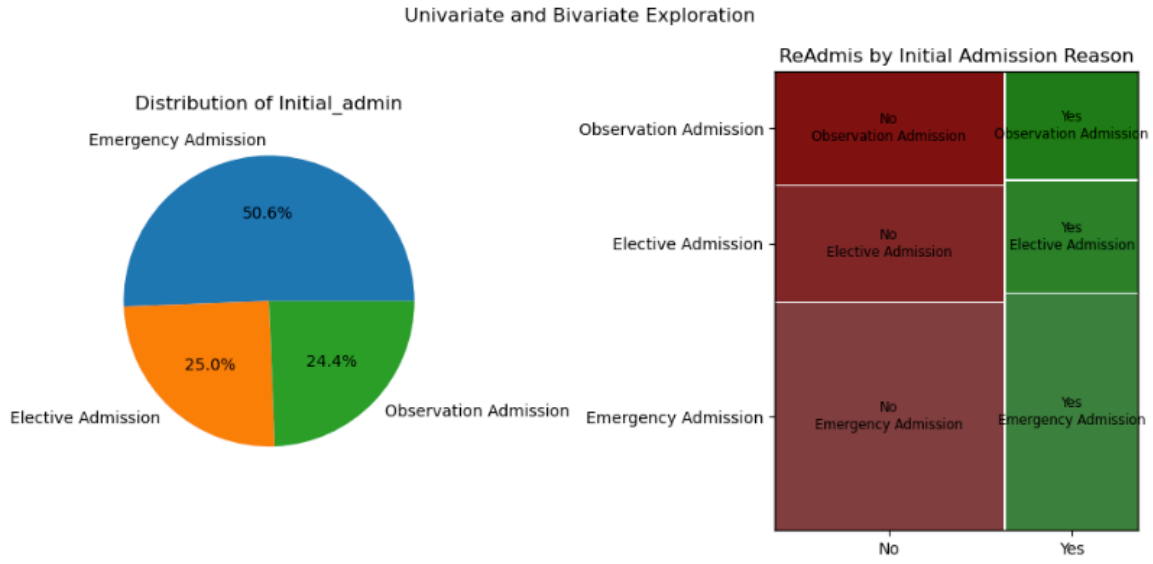
The chart below shows the distribution of the independent variable ‘Initial_admin’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Initial_admin’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show the distribution of the reason for being admitted, the reason, and then the dispersion inside, whether it was readmission or not. The charts do not show any distinct correlation between the fields. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Initial_admin Dependent Variable: ReAdmis
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_ylabel("")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Initial_admin (categorical)
mosaic(df, ['ReAdmis', 'Initial_admin'], ax=axes[1])
axes[1].set_title("ReAdmis by Initial Admission Reason")

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



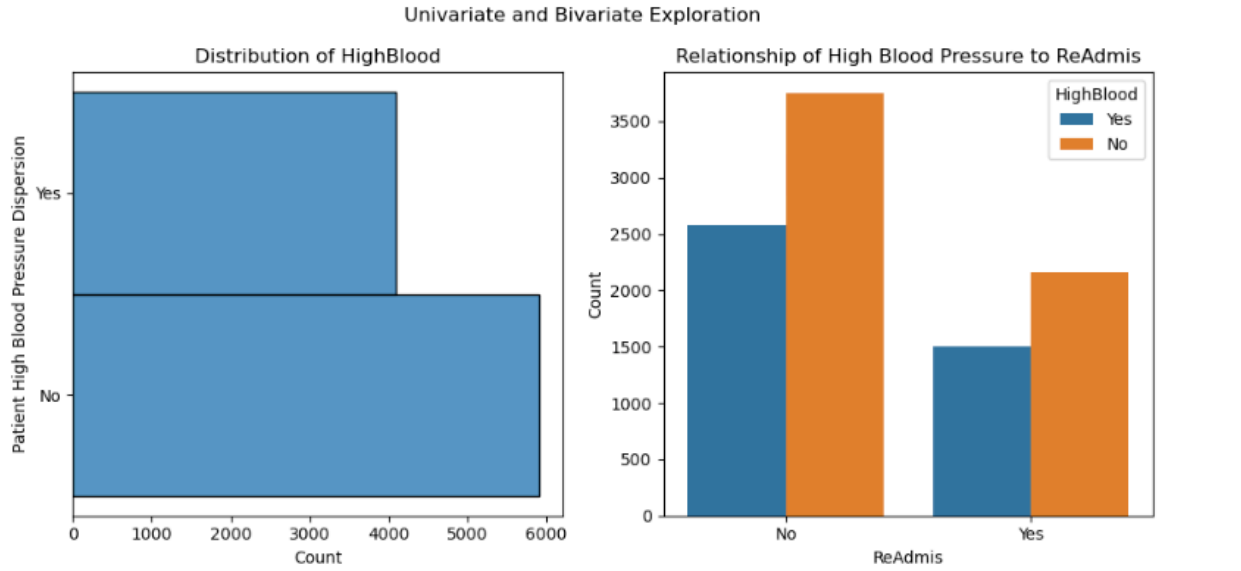
The chart below shows the distribution of the independent variable ‘HighBlood’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘HighBlood’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show an understandable dispersion between the fields based on the number of patients with high blood pressure is not, and readmission, not showing any obvious relationship between the two. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: HighBlood Dependent Variable: ReAdmis
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.
# Hist Chart Analysis on HighBlood
sns.histplot(data=df, y='HighBlood',ax=axes[0])
axes[0].set_title('Distribution of HighBlood')
axes[0].set_ylabel("Patient High Blood Pressure Dispersion")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & High Blood (categorical)
sns.countplot(x="ReAdmis", hue="HighBlood", data=df, ax=axes[1])
axes[1].set_title("Relationship of High Blood Pressure to ReAdmis")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")

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



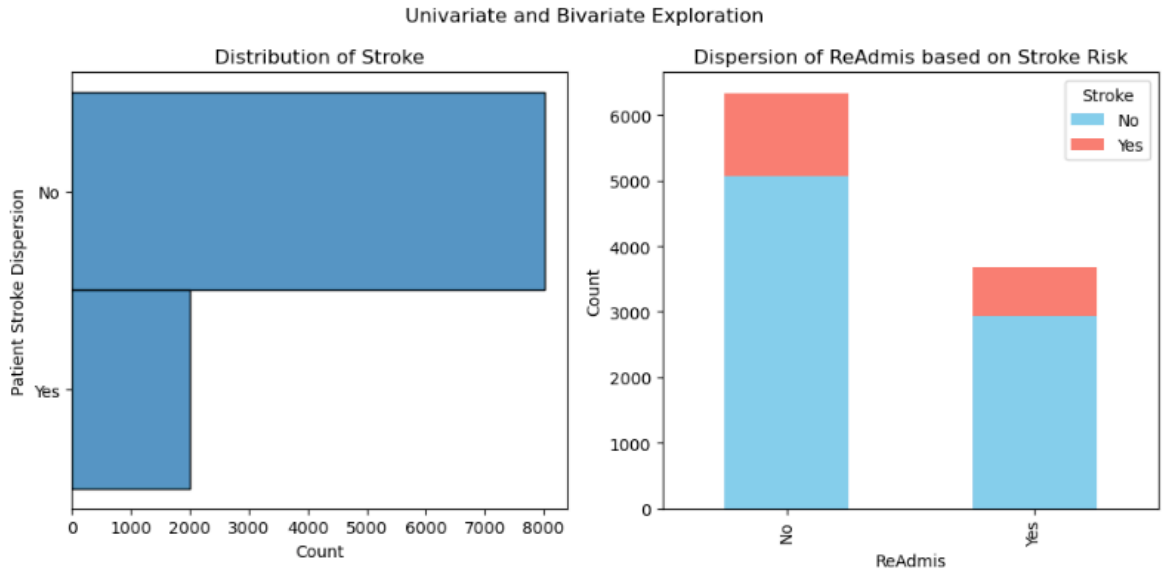
The chart below shows the distribution of the independent variable ‘Stroke’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Stroke’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show more patients have not had a stroke they had. The readmission categories have a normal dispersion based on the number of patients. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Stroke Dependent Variable: ReAdmis
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.
# Hist Chart Analysis on Stroke
sns.histplot(data=df, y='Stroke',ax=axes[0])
axes[0].set_title('Distribution of Stroke')
axes[0].set_ylabel("Patient Stroke Dispersion")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Stroke (categorical)
crosstab_data = pd.crosstab(df['ReAdmis'], df['Stroke'])
crosstab_data.plot(kind='bar', stacked=True, ax=axes[1], color=['skyblue', 'salmon'])
axes[1].set_title("Dispersion of ReAdmis based on Stroke Risk")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")

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



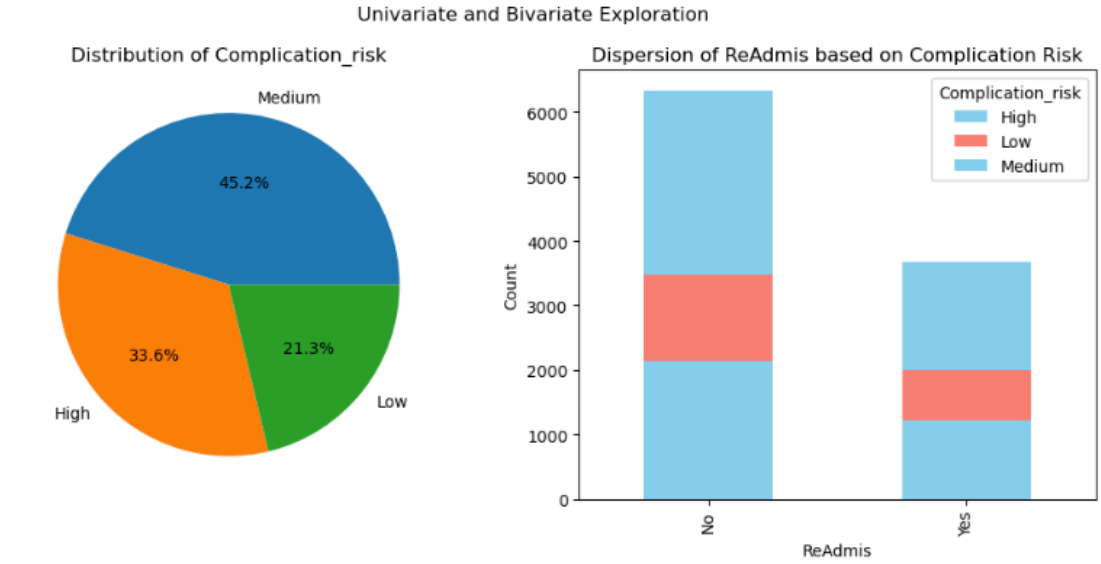
The chart below shows the distribution of the independent variable 'Complication_risk' using a Univariate analysis, as well as the distribution and correlation between the independent variable 'Complication_risk' and the dependent variable 'ReAdmis' using a Bivariate Analysis. The visualizations below show the distribution of complication risk and then the dispersion of where those categories fall into the readmission categories. It looks like there is an even dispersion across the fields based on the number of patients. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Complication_risk Dependent Variable: ReAdmis
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 Complication_risk
df['Complication_risk'].value_counts().plot.pie(autopct='%1.1f%%', ax=axes[0])
axes[0].set_title('Distribution of Complication_risk')
axes[0].set_ylabel("")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Complication_risk (categorical)
crosstab_data = pd.crosstab(df['ReAdmis'], df['Complication_risk'])
crosstab_data.plot(kind='bar', stacked=True, ax=axes[1], color=['skyblue', 'salmon'])
axes[1].set_title("Dispersion of ReAdmis based on Complication Risk")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")

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



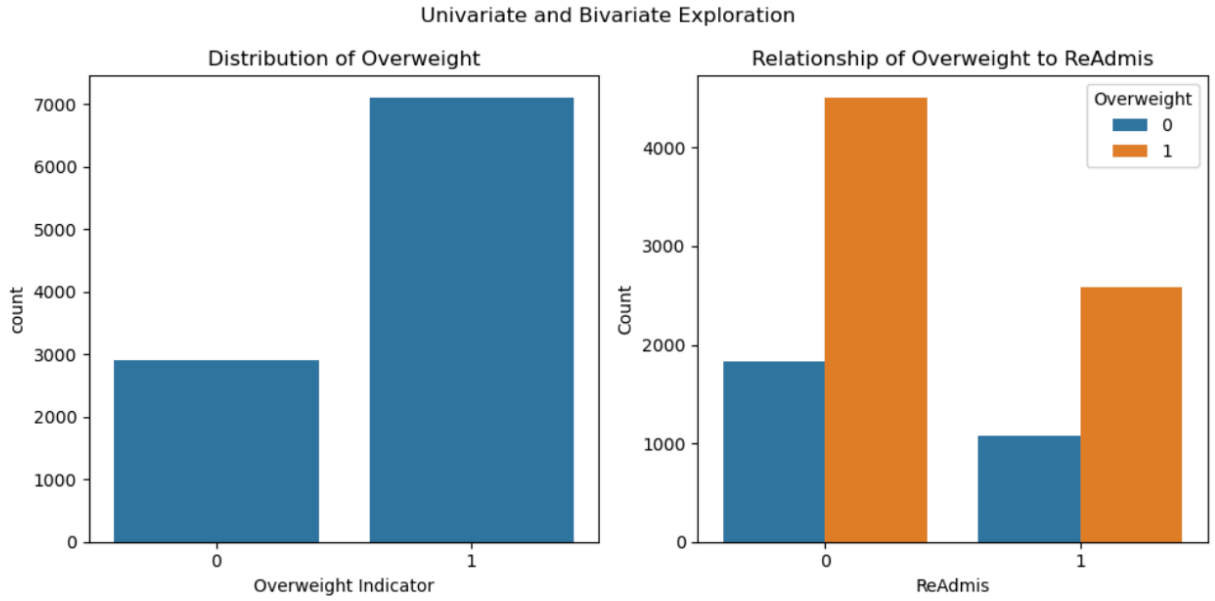
The chart below shows the distribution of the independent variable ‘Overweight’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Overweight’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show that fewer patients are overweight. Overweight patients are the higher count in the readmission categories as well. However, the dispersion appears even and would require additional analysis to confirm any correlation between the two variables. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent
# Independent Variable: Overweight Dependent Variable: ReAdmis
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 Overweight
sns.countplot(data=df, x='Overweight', ax=axes[0])
axes[0].set_title('Distribution of Overweight')
axes[0].set_xlabel("Count")
axes[0].set_xlabel("Overweight Indicator")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Overweight (categorical)
sns.countplot(x="ReAdmis", hue="Overweight", data=df, ax=axes[1])
axes[1].set_title("Relationship of Overweight to ReAdmis")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")

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



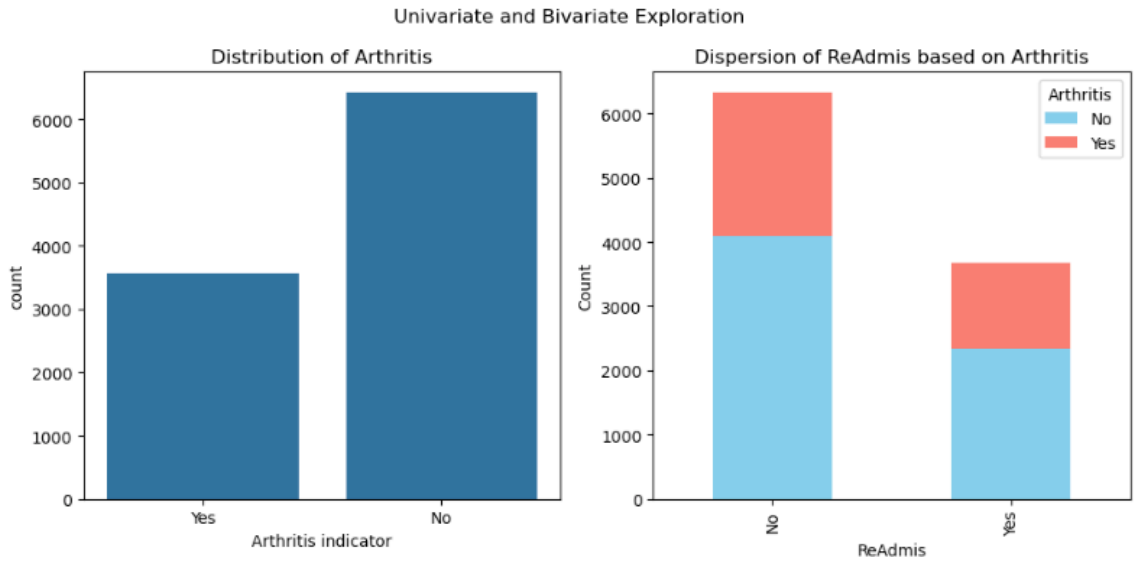
The chart below shows the distribution of the independent variable ‘Arthritis’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Arthritis’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show there are fewer patients with arthritis than not, and there is a normal dispersion between that and patient readmissions, indicating no obvious relationship between the variables. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Arthritis Dependent Variable: ReAdmis
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_xlabel("Count")
axes[0].set_xlabel("Arthritis indicator")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Arthritis (categorical)
crosstab_data = pd.crosstab(df['ReAdmis'], df['Arthritis'])
crosstab_data.plot(kind='bar', stacked=True, ax=axes[1], color=['skyblue', 'salmon'])
axes[1].set_title("Dispersion of ReAdmis based on Arthritis")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")

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



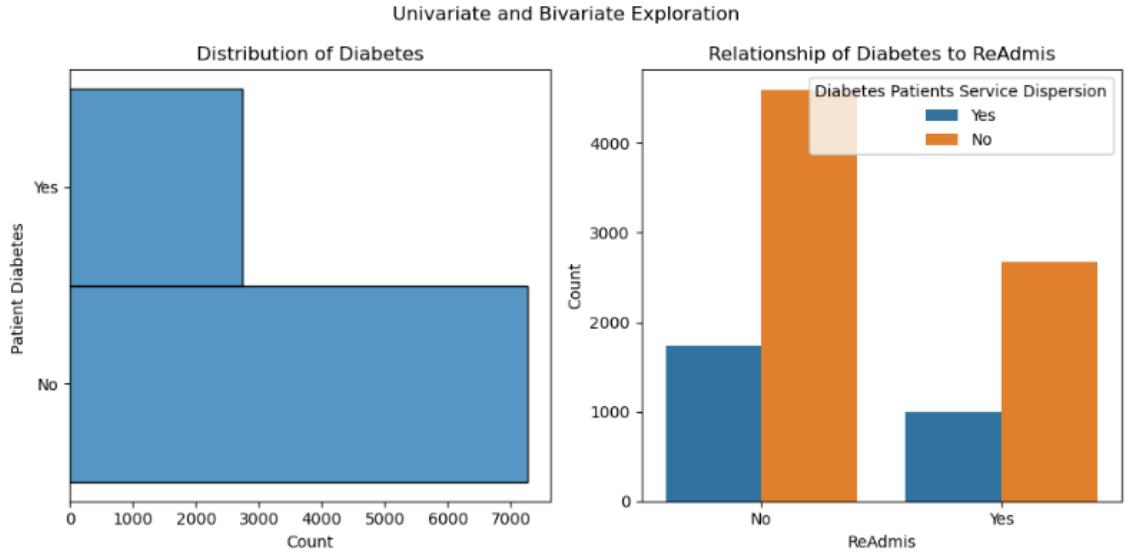
The chart below shows the distribution of the independent variable ‘Diabetes’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Diabetes’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show the dispersion of patients with diabetes among those who have been readmitted or not. More patients do not have diabetes, and more patients have not been readmitted. Indicating no distinct correlation between the variables. Additional analysis is needed to determine the relationship between the variables. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Diabetes Dependent Variable: ReAdmis
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.
# Hist Plot Analysis on Diabetes
sns.histplot(data=df, y='Diabetes',ax=axes[0])
axes[0].set_title('Distribution of Diabetes')
axes[0].set_ylabel("Patient Diabetes ")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Diabetes (categorical)
sns.countplot(x="ReAdmis", hue="Diabetes", data=df, ax=axes[1])
axes[1].set_title("Relationship of Diabetes to ReAdmis")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")
axes[1].legend(title="Diabetes Patients Service Dispersion")

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



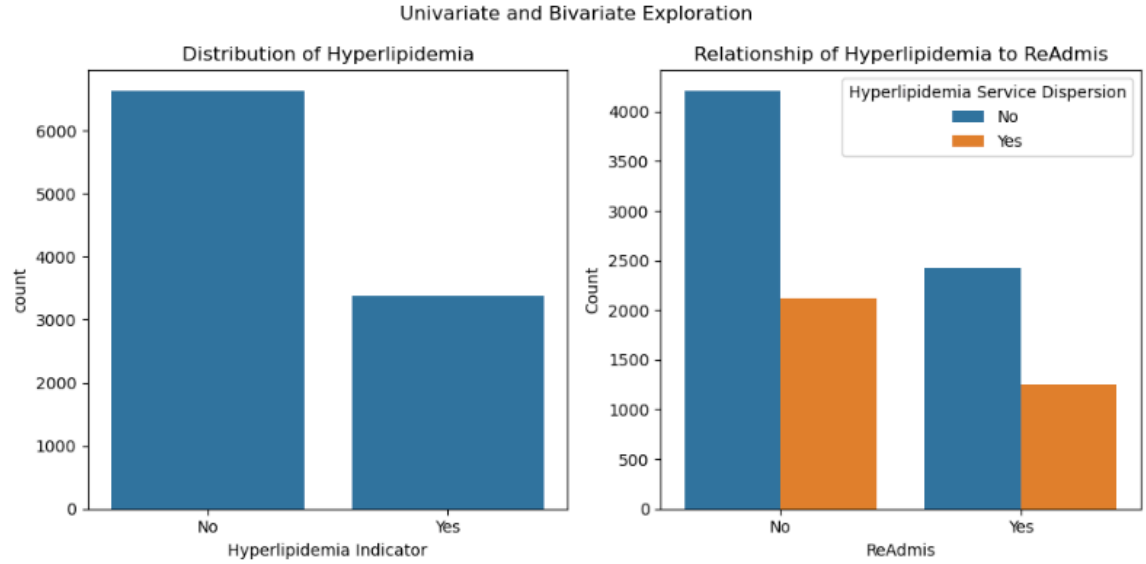
The chart below shows the distribution of the independent variable ‘Hyperlipidemia’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Hyperlipidemia’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show the dispersion of patients with hyperlipidemia against hospital readmission. This indicates there are readmissions with patients who do not have hyperlipidemia. The number of patients who do not have hyperlipidemia is higher than those who do, indicating there is not a strong relationship between the variables. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Hyperlipidemia Dependent Variable: ReAdmis
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 Hyperlipidemia
sns.countplot(data=df, x='Hyperlipidemia', ax=axes[0])
axes[0].set_title('Distribution of Hyperlipidemia')
axes[0].set_xlabel("Count")
axes[0].set_xlabel("Hyperlipidemia Indicator")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Hyperlipidemia (categorical)
sns.countplot(x="ReAdmis", hue="Hyperlipidemia", data=df, ax=axes[1])
axes[1].set_title("Relationship of Hyperlipidemia to ReAdmis")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")
axes[1].legend(title="Hyperlipidemia Service Dispersion")

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



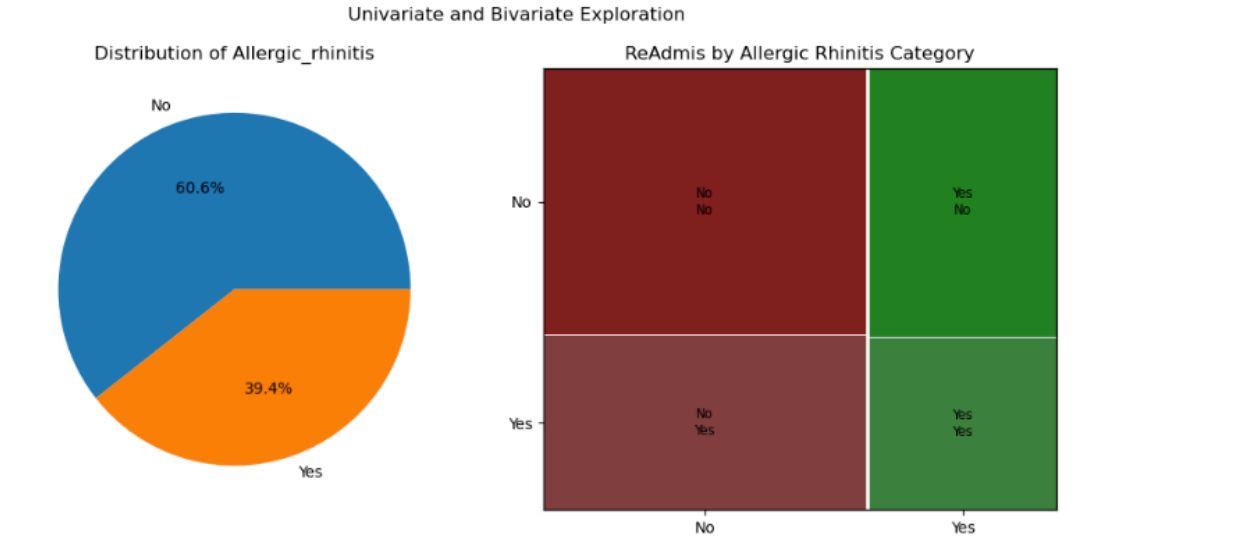
The chart below shows the distribution of the independent variable ‘Allergic_rhinitis’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Allergic_rhinitis’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show the dispersion of rhinitis and hospital readmissions. The charts do not indicate any distinct relationship between the variables. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Allergic_rhinitis Dependent Variable: ReAdmis
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 Allergic_rhinitis
df['Allergic_rhinitis'].value_counts().plot.pie(autopct='%1.1f%%', ax=axes[0])
axes[0].set_title('Distribution of Allergic_rhinitis')
axes[0].set_ylabel("")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Allergic_rhinitis (categorical)
mosaic(df, ['ReAdmis', 'Allergic_rhinitis'], ax=axes[1])
axes[1].set_title("ReAdmis by Allergic Rhinitis Category")

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



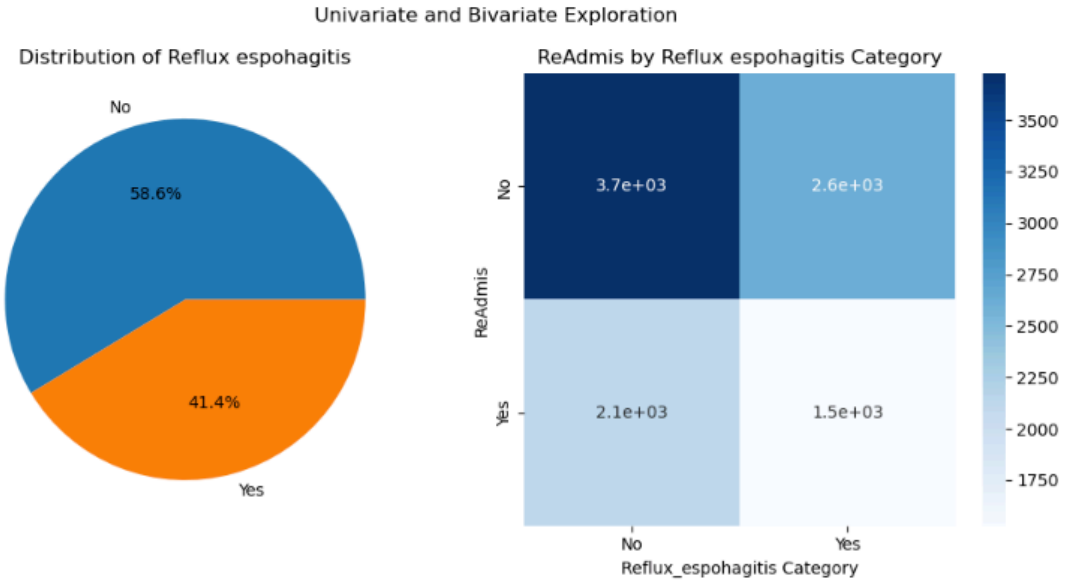
The chart below shows the distribution of the independent variable ‘Reflux_esophagitis’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Reflux_espohagitis’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show the dispersion of reflux. The charts do not indicate any distinct relationship between the variables. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Reflux_espohagitis  Dependent Variable: ReAdmis
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 Reflux_espohagitis
df['Reflux_esophagitis'].value_counts().plot.pie(autopct='%1.1f%%', ax=axes[0])
axes[0].set_title('Distribution of Reflux espohagitis')
axes[0].set_ylabel("")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Reflux_espohagitis (categorical)
crosstab_data = pd.crosstab(df['ReAdmis'], df['Reflux_esophagitis'])
sns.heatmap(crosstab_data, annot=True, cmap="Blues", ax=axes[1])
axes[1].set_title("ReAdmis by Reflux espohagitis Category")
axes[1].set_xlabel("Reflux_espohagitis Category")
axes[1].set_ylabel("ReAdmis")

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



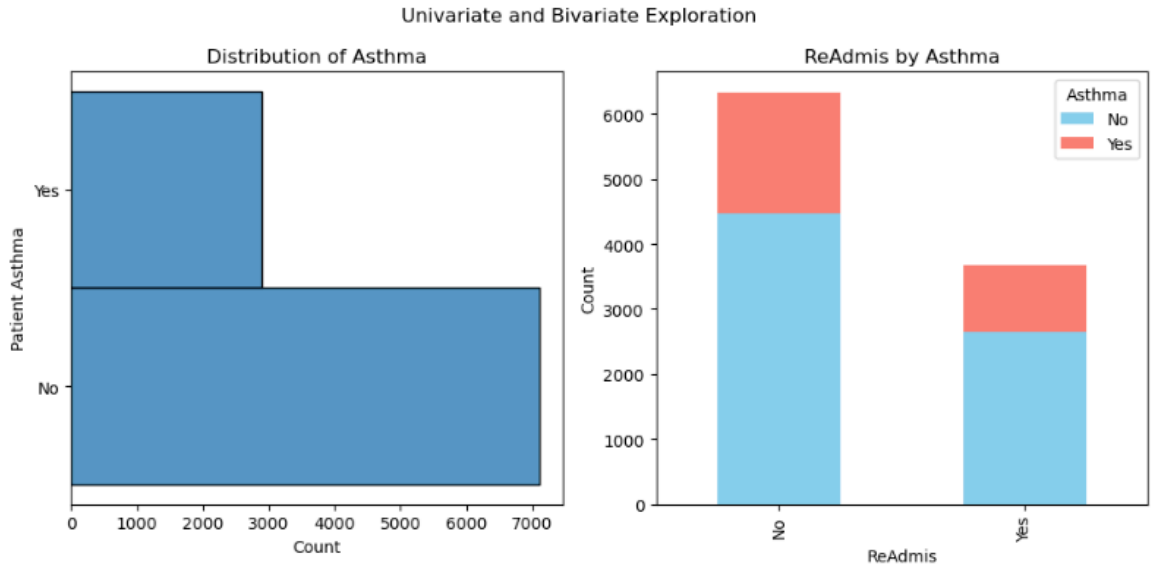
The chart below shows the distribution of the independent variable ‘Asthma’ using a Univariate analysis, as well as the distribution and correlation between the independent variable ‘Asthma’ and the dependent variable ‘ReAdmis’ using a Bivariate Analysis. The visualizations below show the dispersion of asthma against hospital readmissions. The charts show more patients have been readmitted who do not have asthma. However, there are a higher number of patients without asthma, therefore evidencing no correlation between the two variables. Additional analysis is needed to determine the relationship between the variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Asthma  Dependent Variable: ReAdmis
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.
# Hist Plot Analysis on Asthma
sns.histplot(data=df, y='Asthma',ax=axes[0])
axes[0].set_title('Distribution of Asthma')
axes[0].set_ylabel("Patient Asthma")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Asthma (categorical)
crosstab_data = pd.crosstab(df['ReAdmis'], df['Asthma'])
crosstab_data.plot(kind='bar', stacked=True, ax=axes[1], color=['skyblue', 'salmon'])
axes[1].set_title("ReAdmis by Asthma")
axes[1].set_xlabel("ReAdmis")
axes[1].set_ylabel("Count")

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



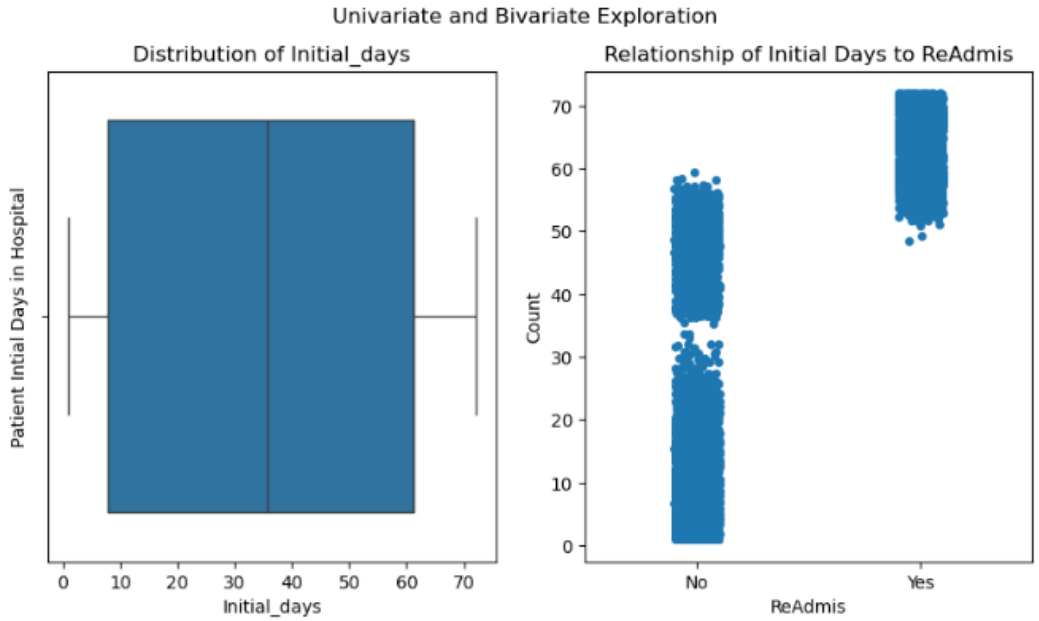
The chart below shows the distribution of the independent variable 'Initial_days' using a Univariate analysis, as well as the distribution and correlation between the independent variable 'Initial_days' and the dependent variable 'ReAdmis' using a Bivariate Analysis. The visualizations below show how the relationship between the initial days in the hospital and readmissions appears to be a normal dispersion across the categories. These charts do not indicate a distinct linear relationship between the variables. Additional analysis should be done to determine the correlation between the two variables.

```
# Performed Univariate on one independent variable and Bivariate analysis on the independent and dependent variable.
# Independent Variable: Initial_days Dependent Variable: ReAdmis
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.
# Hist Plot Analysis on Initial_days
sns.boxplot(data=df, x='Initial_days',ax=axes[0])
axes[0].set_title('Distribution of Initial_days')
axes[0].set_ylabel("Patient Intial Days in Hospital")

# RIGHT plot: Bivariate exploration of ReAdmis (categorical) & Initial_days (continuous)
sns.stripplot(x="ReAdmis", y="Initial_days", data=df, jitter=True)
plt.title("Relationship of Initial Days to ReAdmis")
plt.xlabel("ReAdmis")
plt.ylabel("Count")
plt.show()

# 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 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                 Yes
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 six 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. Job, Time zone, State, and City all also have high cardinality, and we will remove these. 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. Lastly, a few categorical values (including back pain, anxiety, marital, etc.) were not included due to industry knowledge and the assumption that these would not provide value.

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)
- ReAdmis (Category)
- Soft_drink (Category)
- Initial_admin (Category)
- HighBlood (Category)
- Stroke (Category)
- Complication_risk (Category)
- Services (Category)
- Overweight (Category)
- Arthritis (Category)
- Diabetes (Category)
- Hyperlipidemia (Category)
- Allergic_rhinitis (Category)
- Reflux_esophagitis (Category)
- Asthma (Category)

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 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.

```
# Reduce the number of levels (categories) that will be kept in the model to simplify it.
# Using get dummies on variables ["Gender","ReAdmis","Soft_drink","Initial_admin","HighBlood","Stroke","Complication_risk","Overweight","Arthritis","Diabetes","Hyperlipidemia","Allergic_rhinitis","Refl
columns_for_dummy = ["Gender","Services","Soft_drink","Initial_admin","HighBlood","Stroke","Complication_risk","Overweight","Arthritis","Diabetes","Hyperlipidemia","Allergic_rhinitis","Reflux_esophagi
data_dummies = pd.get_dummies(data=df[columns_for_dummy], drop_first=True)
```

```
# Combine variables to create new DF with variables we are interested in using. Excluding the Dependent variable of "TotalCharge"
start_df = df[["Age","Doc_visits","Initial_days"]]
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)
```

```
print(model_df)
```

```
           Age  Doc_visits  Initial_days  Gender_Male  Gender_Nonbinary  \
CaseOrder
1           53           6            10            1                 0
2           51           4            15            0                 0
3           53           4             4            0                 0
4           78           4             1            1                 0
5           22           5             1            0                 0
...        ...         ...           ...          ...               ...
9996        25           4            51            1                 0
9997        87           5            68            1                 0
9998        45           4            70            0                 0
9999        43           5            63            1                 0
10000       70           5            70            0                 0

           Services_CT Scan  Services_Intravenous  Services_MRI  Soft_drink_1  \
CaseOrder
1                         0                     0             0             0
2                         0                     1             0             0
3                         0                     0             0             0
4                         0                     0             0             0
5                         1                     0             0             1
...                     ...                   ...           ...           ...
9996                      0                     1             0             0
9997                      1                     0             0             0
9998                      0                     1             0             1
9999                      0                     0             0             0
10000                     0                     0             0             0

           Initial_admin_Emergency Admission  ...  Stroke_1  \
CaseOrder                                     ...
1                                          1  ...         0
2                                          1  ...         0
3                                          0  ...         0
4                                          0  ...         1
5                                          0  ...         0
...                                      ...  ...       ...
9996                                       1  ...         0
9997                                       0  ...         0
9998                                       0  ...         0
9999                                       1  ...         0
10000                                      0  ...         0

           Complication_risk_Low  Complication_risk_Medium  Overweight_1  \
CaseOrder
1                              0                         1             0
2                              0                         0             1
3                              0                         1             1
4                              0                         1             0
5                              1                         0             0
...                          ...                       ...           ...
9996                           0                         1             0
9997                           0                         1             1
9998                           0                         0             1
9999                           0                         1             1
10000                          1                         0             1

           Arthritis_1  Diabetes_1  Hyperlipidemia_1  Allergic_rhinitis_1  \
CaseOrder
1                    1           1                 0                    1
2                    0           0                 0                    0
3                    0           1                 0                    0
4                    1           0                 0                    0
5                    0           0                 1                    1
...                ...         ...               ...                  ...
9996                 0           0                 0                    0
9997                 1           1                 0                    0
9998                 0           0                 0                    1
9999                 0           0                 0                    0
10000                1           0                 1                    1

           Reflux_esophagitis_1  Asthma_1
CaseOrder
1                             0         1
2                             1         0
3                             0         0
4                             1         1
5                             0         0
...                         ...       ...
9996                          1         0
9997                          0         1
9998                          0         0
9999                          0         0
10000                         0         0

[10000 rows x 22 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, it 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)
# Low VIF indicating no distinct linear relationship between variables that could cause instability in the model.
model_df = model_df.apply(pd.to_numeric, errors='coerce')
model_df = model_df.dropna()
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	46.080779
1	Age	1.002059
2	Doc_visits	1.002888
3	Initial_days	1.002085
4	Gender_Male	1.022772
5	Gender_Nonbinary	1.022307
6	Services_CT Scan	1.083661
7	Services_Intravenous	1.097132
8	Services_MRI	1.033646
9	Soft_drink_1	1.002150
10	Initial_admin_Emergency Admission	1.495391
11	Initial_admin_Observation Admission	1.497616
12	HighBlood_1	1.002517
13	Stroke_1	1.001485
14	Complication_risk_Low	1.288429
15	Complication_risk_Medium	1.288973
16	Overweight_1	1.002003
17	Arthritis_1	1.002106
18	Diabetes_1	1.002041
19	Hyperlipidemia_1	1.002664
20	Allergic_rhinitis_1	1.001996
21	Reflux_esophagitis_1	1.002386
22	Asthma_1	1.001890

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 somewhat high (46.08), 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

around 1, 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_task2.csv' file.

D, Model Comparison

An initial and reduced logistic regression model analysis was performed to model the relationship between a categorical 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 Logistic Regression Model

An initial multiple logistic 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 logistic regression model. A constant was added, which includes all the independent variables, and then the model was fit to learn the variables.

Logistic regression focuses on the probability of one of the two categories in a binary outcome. In this case, the probability output of readmission occurring (or ReAdmis = 1) is the outcome we are interested in. The probability of ReAdmis = 0 (no readmission) can be derived as the complement to the probability of ReAdmis = 1.

The results of the regression model will be further assessed to ensure all independent variables included contribute meaningful value to the dependent variable.

```
# Define dependent and independent variables
y = df["ReAdmis"]
X = model_df

# Add constant to the independent variables
X = sm.add_constant(X)

# Build logistic regression model
logit_model = sm.MNLogit(y, X)
result = logit_model.fit()

# Print summary of the model
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.035813
         Iterations 14
                          MNLogit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                        MNLogit   Df Residuals:                     9977
Method:                           MLE   Df Model:                           22
Date:                Wed, 02 Oct 2024   Pseudo R-squ.:                 0.9455
Time:                        11:13:03   Log-Likelihood:                -358.13
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==========================================================================================================
                         ReAdmis=1       coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------------------
const                                -75.1752      4.103    -18.321      0.000     -83.217     -67.133
Age                                    0.0015      0.005      0.309      0.757      -0.008       0.011
Doc_visits                            -0.0210      0.092     -0.227      0.820      -0.202       0.160
Initial_days                           1.3810      0.075     18.418      0.000       1.234       1.528
Gender_Male                            0.2430      0.198      1.226      0.220      -0.145       0.631
Gender_Nonbinary                       0.7070      0.681      1.038      0.299      -0.628       2.042
Services_CT Scan                       1.5120      0.344      4.401      0.000       0.839       2.185
Services_Intravenous                  -0.0445      0.219     -0.204      0.839      -0.473       0.384
Services_MRI                           2.5896      0.482      5.371      0.000       1.645       3.535
Soft_drink_1                           0.2750      0.228      1.207      0.227      -0.171       0.722
Initial_admin_Emergency Admission      2.4524      0.271      9.058      0.000       1.922       2.983
Initial_admin_Observation Admission    0.8980      0.270      3.330      0.001       0.369       1.427
HighBlood_1                            0.7986      0.204      3.914      0.000       0.399       1.199
Stroke_1                               1.5358      0.256      5.997      0.000       1.034       2.038
Complication_risk_Low                 -1.7490      0.283     -6.174      0.000      -2.304      -1.194
Complication_risk_Medium              -0.4281      0.224     -1.909      0.056      -0.868       0.011
Overweight_1                          -0.2664      0.217     -1.226      0.220      -0.692       0.160
Arthritis_1                           -1.2720      0.215     -5.910      0.000      -1.694      -0.850
Diabetes_1                             0.4675      0.218      2.145      0.032       0.040       0.895
Hyperlipidemia_1                       0.2655      0.207      1.281      0.200      -0.141       0.672
Allergic_rhinitis_1                   -0.3060      0.197     -1.554      0.120      -0.692       0.080
Reflux_esophagitis_1                  -0.2750      0.201     -1.371      0.170      -0.668       0.118
Asthma_1                              -1.3583      0.224     -6.059      0.000      -1.798      -0.919
==========================================================================================================
```

D2, Model Reduction

The above logistic regression model did not produce any warnings that needed to be reviewed. Due to the dependent variable being categorical, it is important to review the various p-values for the intended variables across the dependent categories. Predictors with p values >.05 are considered statistically significant and hurt the probability of the dependent variable.

Additionally, it is required that the independent variables do not have high multicollinearity, to ensure this is the case, the VIF (Variance Inflation Factor) was assessed. The proper rule 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 Doc_visits was 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 Doc_visits from the data frame.

```
# Check for VIF to see if variables should be eliminated due to high multicolinearity
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	6.632703
1	Doc_visits	12.134501
2	Initial_days	2.574981
3	Gender_Male	1.903299
4	Gender_Nonbinary	1.042636
5	Services_CT Scan	1.228136
6	Services_Intravenous	1.568013
7	Services_MRI	1.071192
8	Soft_drink_1	1.341591
9	Initial_admin_Emergency Admission	2.853776
10	Initial_admin_Observation Admission	1.900625
11	HighBlood_1	1.676874
12	Stroke_1	1.243213
13	Complication_risk_Low	1.593602
14	Complication_risk_Medium	2.263677
15	Overweight_1	3.276016
16	Arthritis_1	1.545015
17	Diabetes_1	1.371704
18	Hyperlipidemia_1	1.490966
19	Allergic_rhinitis_1	1.630818
20	Reflux_esophagitis_1	1.674043
21	Asthma_1	1.395590

```
]: # Removing Doc_visits and Initial_days from model_df due to high multicollinearity
   model_df = model_df.drop(columns=['Doc_visits'])
```

After dropping Doc_visits 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 5.48, which is slightly higher compared to the others. Due to the score, these should be monitored but not removed. No other variables returned a high multicollinearity value.

```
# 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	5.485368
1	Initial_days	2.488877
2	Gender_Male	1.857135
3	Gender_Nonbinary	1.040826
4	Services_CT Scan	1.218238
5	Services_Intravenous	1.544261
6	Services_MRI	1.069777
7	Soft_drink_1	1.329621
8	Initial_admin_Emergency Admission	2.649686
9	Initial_admin_Observation Admission	1.796076
10	HighBlood_1	1.653448
11	Stroke_1	1.236648
12	Complication_risk_Low	1.559522
13	Complication_risk_Medium	2.193198
14	Overweight_1	3.080646
15	Arthritis_1	1.531188
16	Diabetes_1	1.359200
17	Hyperlipidemia_1	1.481132
18	Allergic_rhinitis_1	1.606674
19	Reflux_esophagitis_1	1.643014
20	Asthma_1	1.388205

D3, Reduced Logistic Regression Model

After an initial logistic 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.

The first step in reducing the model was to initialize the reduction by using a feature selection and RFE. The number of features selected to be included in the RFE output came from the results of the initial model. There were 11 variables with p-values <.05. Thus, 11 variables were inputted to RFE. A feature ranting was outputted, showing the selected features rank. After reduction techniques have been performed, the new variable selection can be inputted into a new data frame called 'X-final.' Summary statistics on the reduced model can be seen below.

```
# Initialize Reduced Logistic Regression model
X_reduced = model_df
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_reduced)

# Step 2: Apply RFE for feature selection with increased max_iter and different solver
model = LogisticRegression(max_iter=1000, solver='liblinear')
rfe = RFE(estimator=model, n_features_to_select=11)
rfe = rfe.fit(X_scaled, y)

# Check selected features
selected_features = X_reduced.columns[rfe.support_]
feature_ranking = rfe.ranking_

print("Selected Features:", selected_features)
print("Feature Ranking:", feature_ranking)
```

```
Selected Features: Index(['Initial_days', 'Services_CT Scan', 'Services_MRI',
       'Initial_admin_Emergency Admission',
       'Initial_admin_Observation Admission', 'HighBlood_1', 'Stroke_1',
       'Complication_risk_Low', 'Arthritis_1', 'Diabetes_1', 'Asthma_1'],
      dtype='object')
Feature Ranking: [ 8  1  4  6  1 11  1  9  1  1  1  1  1  3 10  1  1  5  2  7  1]
```

```
X_final = pd.DataFrame(X_scaled, columns=X_reduced.columns)[selected_features]

# Align indices of X_final and y if necessary
X_final.reset_index(drop=True, inplace=True)
y.reset_index(drop=True, inplace=True)

# Step 4: Fit the reduced logistic regression model with selected features
logit_model_final = sm.MNLogit(y, sm.add_constant(X_final))
result_final = logit_model_final.fit()

# Print summary of the reduced model
print(result_final.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.036592
         Iterations 14
                          MNLogit Regression Results
==============================================================================
Dep. Variable:                 ReAdmis   No. Observations:                10000
Model:                         MNLogit   Df Residuals:                     9988
Method:                            MLE   Df Model:                           11
Date:                Thu, 03 Oct 2024   Pseudo R-squ.:                  0.9443
Time:                        15:46:48   Log-Likelihood:                -365.92
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                         ReAdmis=1       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                                -26.6218      1.422    -18.724      0.000     -29.409     -23.835
Initial_days                          35.3364      1.875     18.851      0.000      31.662      39.010
Services_CT Scan                       0.4884      0.108      4.538      0.000       0.277       0.699
Services_MRI                           0.4920      0.090      5.447      0.000       0.315       0.669
Initial_admin_Emergency Admission      1.1607      0.130      8.938      0.000       0.906       1.415
Initial_admin_Observation Admission    0.3484      0.113      3.075      0.002       0.126       0.570
HighBlood_1                            0.3624      0.098      3.700      0.000       0.170       0.554
Stroke_1                               0.5955      0.100      5.930      0.000       0.399       0.792
Complication_risk_Low                 -0.6010      0.099     -6.056      0.000      -0.796      -0.407
Arthritis_1                           -0.5821      0.101     -5.767      0.000      -0.780      -0.384
Diabetes_1                             0.2065      0.096      2.161      0.031       0.019       0.394
Asthma_1                              -0.5916      0.100     -5.941      0.000      -0.787      -0.396
================================================================================================
```

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 logistic 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, ReAdmis. 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 hospital readmissions.

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—Accuracy output, log-loss, and confusion matrix.

A model evaluation metric was created to output results to compare the initial multiple logistic regression model with the reduced logistic regression model. Model evaluation metrics include the Accuracy calculation output, log-loss, and confusion matrix.

Full model metrics -

```
# FULL Model Evaluation Metrics - Accuracy, Log Loss, Confusion Matrix
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build the logistic regression model
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train, y_train)

# Predict on the test set
y_pred = logreg.predict(X_test)
y_pred_prob = logreg.predict_proba(X_test)[:, 1]

# Evaluate model performance
# Accuracy
accuracy = accuracy_score(y_test, y_pred)

# Log Loss
logloss = log_loss(y_test, y_pred_prob)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Step 6: Print the metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Log Loss: {logloss:.4f}")
print("\nConfusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.9865
Log Loss: 0.0344

Confusion Matrix:
[[1284    7]
 [  20  689]]
```

Reduced model metrics –

```
# REDUCED Model Evaluation Metrics - Accuracy, Log Loss, Confusion Matrix
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_final, y, test_size=0.2, random_state=42)

# Build the logistic regression model
logreg_reduced = LogisticRegression(max_iter=1000)
logreg_reduced.fit(X_train, y_train)

# Predict on the test set
y_pred = logreg_reduced.predict(X_test)
y_pred_prob = logreg_reduced.predict_proba(X_test)[:, 1]

# Evaluate model performance
# Accuracy
accuracy = accuracy_score(y_test, y_pred)

# Log Loss
logloss = log_loss(y_test, y_pred_prob)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Step 6: Print the metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Log Loss: {logloss:.4f}")
print("\nConfusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.9850
Log Loss: 0.0488

Confusion Matrix:
[[1277   14]
 [  16  693]]
```

The Accuracy number on the full model came out to .9865, while the reduced model was slightly lower at .9850. This indicates that the full model has a slightly better prediction rate, with the numbers being very close. The full model has a log loss of .0344, and the reduced has a log loss of .0488. Indicating that the full model is slightly better at predicting probabilities. Lastly, the confusion matrix helps to determine how well the model is predicting the classes. This is explained under E2 below.

Both models have very high accuracy (98.65% and 98.50%), which suggests that the models are performing well on the test data, indicating a good initial sign of model performance.

The Log Loss values are quite low (0.0344 and 0.0488), which indicates that the models' probability predictions are close to the true labels. Lower Log Loss means better-calibrated probabilities. A value close to 0 is what we expect from a well-performing model.

The first model has slightly better performance on accuracy and log loss compared to the second. The differences in confusion matrix elements are also minor, which indicates that both models are functioning well and producing relatively similar outcomes.

In conclusion, both models have very high accuracy and low log loss, and the confusion matrix shows that they are classifying most cases correctly. Both models, however, may have limitations in terms of their explanatory power, suggesting that further refinement or additional modeling approaches might be necessary.

E2, Model Output

The output and calculations of the analysis performed, including an accuracy calculation and a confusion matrix, are shown below. The code was combined with the metrics and predictions printed under section E1 above, posting the outputted results of these for discussion here.

```
Accuracy: 0.9865
Log Loss: 0.0344

Confusion Matrix:
[[1284    7]
 [  20  689]]
```

Full model -

```
Accuracy: 0.9850
Log Loss: 0.0488

Confusion Matrix:
[[1277   14]
 [  16  693]]
```

Reduced model -

An accuracy metric is used to evaluate classification models and how well they are performing. It is defined as the ratio of correctly predicted observations to total observations. The following formula can be used to calculate the accuracy:

Accuracy = Number of Correct Predictions / Total Number of Predictions

A confusion matrix can also be used to evaluate the performance of a model. It compares the actual labels to the predicted labels, and it provides insights into the types of errors the model is making.

For the full model, the overall accuracy of 98.65% indicates the model is correctly predicting the label 98.65% of the time. The confusion matrix indicates that it is classifying most cases correctly. For the reduced model, the overall accuracy of 98.50% indicates the model is correctly predicting the label 98.50% of the time. The confusion matrix indicates that it is classifying most cases correctly.

Overall, the evaluation of both logistic regression models (initial and reduced) shows strong performance. However, given that the initial model has slightly better log loss and fewer false positives, it may be the preferred model. Both models demonstrate strong predictive capability, and the reduced model may still be valuable if feature reduction is important for simplifying the model while maintaining high performance.

E3, Model Code

An executable error-free copy of the code used to support the implementation of the logistic regression models performed in Python can be found within the attached 'D208-Code-Task2' 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:
$\text{Log} = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta nXn$

- Initial_days (35.3364): For each additional day in the hospital, the log odds of being readmitted increase significantly. This large positive coefficient suggests that longer hospital stays are a strong predictor of readmission.
- Services_CT Scan (0.4884): Patients who undergo a CT scan have an increased log odds of being readmitted.

- Services_MRI (0.4920): Patients who get an MRI scan have increased log odds of being readmitted.
- Initial_admin_Emergency Admission (1.1607): Patients who are admitted through emergency services have much higher log odds of being readmitted compared to others.
- Initial_admin_Observation Admission (.3484): Patients who are admitted through observation services have a slight increase in log odds of being readmitted.
- HighBlood_1 (.3484): Patients who have high blood pressure have a slight increase in log odds of being readmitted.
- Stroke_1 (.5955): Patients who have had a stroke have an increased log odds of being readmitted.
- Complication_risk_Low (-0.6210): Patients classified as having a low complication risk are less likely to be readmitted. The negative coefficient indicates that this variable decreases the log odds of readmission.
- Arthritis_1 (-.5821): Patients who have arthritis have lower log odds of being readmitted compared to those without arthritis.
- Diabetes_1 (.2065): Patients who have diabetes have a slight increase in log odds of being readmitted.
- Asthma_1 (-0.5916): Patients with asthma also have lower log odds of being readmitted compared to patients without asthma.

This equation captures how each of the selected variables influences hospital readmissions. Given the output, the specific equation would look something like this:

Log = −26.6218 + 35.3364 (Initial_days) + 0.4884 (Services_CT Scan) + 0.4920 (Services_MRI) + 1.1607 (Initial_admin_Emergency Admission) + 0.3484 (Initial_admin_Observation Admission) + 0.3624 (HighBlood_1) + 0.5955 (Stroke_1) − 0.6210 (Complication_risk_Low) − 0.5821 (Arthritis_1) + 0.2065 (Diabetes_1) − 0.5916(Asthma_1)

Interpretation of the Coefficients of the Reduced Model

The coefficients in the logistic regression model represent the log odds of the dependent variable (ReAdmis) being where hospital readmission did occur.

- Initial_days (35.3364): For each additional day in the hospital, the log odds of being readmitted increase significantly. This large positive coefficient suggests that longer hospital stays are a strong predictor of readmission.
- Services_CT Scan (0.4884): Patients who undergo a CT scan have an increased log odds of being readmitted.
- Services_MRI (0.4920): Patients who get an MRI scan have increased log odds of being readmitted.
- Initial_admin_Emergency Admission (1.1607): Patients who are admitted through emergency services have much higher log odds of being readmitted compared to others.
- Initial_admin_Observation Admission (.3484): Patients who have been admitted through observation services have a slight increase in log odds of being readmitted.
- HighBlood_1 (.3484): Patients who have high blood pressure have a slight increase in log odds of being readmitted.

- Stroke_1 (.5955): Patients who have had a stroke have increased log odds of being readmitted.
- Complication_risk_Low (-0.6210): Patients classified as having a low complication risk are less likely to be readmitted. The negative coefficient indicates that this variable decreases the log odds of readmission.
- Arthritis_1 (-.5821): Patients who have arthritis have lower log odds of being readmitted compared to those without arthritis.
- Diabetes_1 (.2065): Patients who have diabetes have a slight increase in log odds of being readmitted.
- Asthma_1 (-0.5916): Patients with asthma also have lower log odds of being readmitted compared to patients without asthma

Statistical and Practical Significance of the Reduced Model

The p-values indicate the statistical significance of each coefficient. A p-value less than 0.05 suggests that the variable is statistically significant. In the model, most variables, including Initial_days, Services_CT Scan, Initial_admin_Emergency Admission, Stroke_1, etc., have p-values of less than 0.05, meaning they are statistically significant predictors of readmission.

On the other hand, Arthritis_1, Complication_risk_Low, and Asthma_1, with p-values < 0.05, are statistically significant but have negative coefficients, indicating that they reduce the log odds of readmission.

For Initial_days, the large coefficient for the number of days spent in the hospital has practical significance because it indicates that longer stays strongly predict higher chances of readmission. This has clear implications for patient care management and discharge planning.

For Services_CT Scan and Services_MRI, these variables also suggest that certain procedures or diagnostic services may be markers of more serious conditions or complications that could lead to readmission.

For Complication_risk_Low and Asthma_1, these negative coefficients highlight that certain conditions (e.g., lower-risk patients, asthma) may make a patient less likely to be readmitted, offering insights into which types of patients may require less post-discharge care.

Limitations of the Data Analysis

There are a few limitations of the data analysis to be considered, including while the reduced model simplifies the analysis, it may leave out important variables that could influence readmission risk. Another limitation is the data quality. If the data contains any inconsistencies, missing values, is imbalanced, or is not representative of the overall population, the model's results could be biased. Lastly, Logistic regression assumes a linear relationship between the predictors and the log odds of the outcome, which may not always hold. More complex relationships.

Conclusion

In conclusion, the reduced model shows good statistical significance with high predictive power in terms of hospital readmissions. The practical significance of certain variables like Initial_days, Emergency Admissions, and Services_CT Scan highlights key areas to focus on for improving patient outcomes and reducing readmissions. However, the analysis might benefit from more complex models or interaction terms to capture non-linear effects, and care should be taken regarding the quality of the underlying data.

F2, Course of Action

Given the findings from the reduced regression model, there are quite a few actions we could take to improve the quality of care and reduce readmission costs. One step would be to use the model and determine which variables had the largest impact on hospital readmissions, using this, we could start to develop targeted plans to figure out why this is happening and how to help it. An example would be to target interventions for long hospital stays by implementing early discharge planning and transitional care for patients.

Another actionable step would be to enhance monitoring for patients undergoing specific procedures. Reviewing the procedures that are associated with readmissions could help identify gaps in care and focus efforts on the coordination between inpatient and outpatient services. Lastly, expanding this model while continuously collecting data on hospital readmissions would allow us to reassess the factors contributing to readmissions over time. This will help identify new trends and adjust care strategies as needed.

These actions, based on the significant predictors of readmissions from the logistic regression model, could help improve patient outcomes, reduce hospital readmissions, and enhance the efficiency of hospital care.

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 – Task 2.“

Panopto video link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=ae5ae32e-e3ad-4370-b0f3-b1fe01728ff5

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.