Predictive Modeling Task 2

D208 WGU

TRESA (TESSIE) AUSTIN

Part I: Research Question

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

1. Readmission rates in hospitals are a significant concern as the impact to patient outcomes, costs, and overall quality of care are greatly impacted. A critical objective for any hospital is to reduce readmission rates since it could be tied to regulatory and reimbursement policies. Based on that it's important to research the following question:
   "What factors contribute to a patient being readmitted to the hospital within 30 days of their initial release?"

2. Define the goals of the data analysis.
   If we can determine which demographic, socioeconomic, and clinical factors significantly influence the path to readmission, then hospital administrators can use that data to create actionable plans to reduce rates. Hospitals can shift resources or enhance the initial care of the patient to prevent costly readmissions and ensure patients get the proper treatment the first time in the facility.

Part II: Method Justification

B. Describe logistic regression methods by doing the following:

1. Summarize **four** assumptions of a logistic regression model.
   According to Statisticsoltions.com (2010) logistic regression doesn't make many of the key assumptions of linear regressions particularly regarding linearity, normality, homoscedasticity, and measurement level. There are some assumptions that do apply:
   1. Binary logistic regression requires the observations to be independent of each other.
   2. Binary logistic regression requires the dependent variable to be binary.
   3. Logistic regression requires there to be little or no multicollinearity among the independent variables.
   4. Logistic regression assumes linearity of independent variables and long odds.
2. Describe **two** benefits of using Python or R in support of various phases of the analysis.
   a. Python allows the use of a wide variety of data manipulation using lists, tuples, dictionaries, and sets according to BeyondVerse (2023). The article goes on to say that the use of lists allows versatility, the use of tuples can hold various items but are immutable to ensure data integrity, dictionaries compile collections of values in keyed pairs for efficient data retrieval and sets help to determine and eliminate duplicates since they are unordered collections. These options allow a user to structure and manipulate data easily.
   b. Python also boasts a large variety of libraries for advanced linear regression. According to Luca (2016) Python offers a mature system of packages for data analysis that are very versatile. There are specific packages for commonly used optimization algorithms such as Scikit-learn. Scikit-learn is built on top of SciPy and offers modules for data processing, model selection and validation for data analysis. As a software engineer, Python has translatable syntax and debugging as noted on WGU Information Technology page (nd)

3. Explain why logistic regression is an appropriate technique to analyze "What factors contribute to a patient is readmitted to the hospital within 30 days of their initial release?"

Logistic regression is used when one dependent variable has only two outcomes, such as pass/fail or yes/no or is considered categorical according to Swaminathan (2018). The question above has a dependent variable of readmission within 30 days, which has only two answers, yes or no. Logistic regression also doesn't assume a linear relationship between the

dependent and independent variables so it's suitable for complex scenarios such as the one explored here.

Part III: Data Preparation

C. Summarize the data preparation process for logistic regression by doing the following:

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

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

- An initial check for duplicates and missing values will be done.
- After that inspection of the data types and variables with data will be done.
    - Several data types will need to be changed from their current values to ones that are applicable for logistic regression.
        - Variables that are float64 such as Income, VitD_levels, Initial_days, TotalCharge and Additional_charges will be changed to astype (int)
        - Columns that are yes/no converted to 1/0 such as HighBlood, Stroke, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, and Asthma
        - Initial_admin will need to be adjusted to represent data in a binary numeric form using one hot encoding as described in D208 – Webinar: Getting Started with D208 Part 1 (WGU, 2022). The use of pandas get_dummies() is the best option for completing the needed transformation of data for this project as it takes a series or list and converts each element present to a column heading as it iterates over the data and inserts values of 1 if it's included and 0 if not (Waqar, 2024).

2. Describe the dependent variable and *all* independent variables using summary statistics that are required to answer the research question, including a screenshot of the summary statistics output for each of these variables.

Dependent Variable:

ReAdmis: Whether the patient was readmitted within a month of release or not (yes, no)

Summary Statistics for ReAdmis:

count	10000
unique	2
top	No
freq	6331

Name: ReAdmis, dtype: object

ReAdmis or readmission of the patient has two answers, yes/no. No is the top unique value with 63.31% of patients not being readmitted within 30 days.

Independent Variables:

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

Summary Statistics for Children:

count	10000.000000
mean	2.097200
std	2.163659
min	0.000000
25%	0.000000

50% 1.000000
75% 3.000000
max 10.000000
Name: Children, dtype: float64

The number of children in the household of the patient, regardless of relationship, has a mean of 2.097200 and a median of 1.0000. The maximum number of children reported was 10 and the minimum number is 0.

Age: Age of the patient as reported in admissions information

Summary Statistics for 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
Name: Age, dtype: float64

The age of the patient at time of admission ranges from a minimum of 18 to a maximum of 89. There was a mean of 53.511700 and a median of 53.00000.

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

Summary Statistics for Income:

count 10000.000000
mean 40490.002100
std 28521.152883
min 154.000000
25% 19598.250000
50% 33768.000000
75% 54295.750000
max 207249.000000
Name: Income, dtype: float64

Annual income: of the patient or the primary insurance holder, if they are not the same person ranges from $154.00 to $207,249.00. This is a wide disparity between the minimum and the maximum. The mean is $40,490.00 and the median is $33,768.00.

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

Summary Statistics for VitD_levels:

count 10000.00000
mean 17.46120
std 2.04037
min 9.00000
25% 16.00000
50% 17.00000
75% 19.00000
max 26.00000
Name: VitD_levels, dtype: float64

Patients' vitamin D levels range from 9.00 to 26.00. The mean is 17.46120 with a median of 17.0000.

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

Summary Statistics for 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

Name: Doc_visits, dtype: float64

Visits by the primary physician for each patient ranged from 1 to 9 times. With most patients seeing their doctor between 4-6 times. Doctor visits have a mean of 5.01 and a median of 5, so again data is symmetrically distributed. Up to 75% of the patients saw their doctors 6 times.

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

Summary Statistics for Full_meals_eaten:

count	10000.000000
mean	1.001400
std	1.008117
min	0.000000
25%	0.000000
50%	1.000000
75%	2.000000
max	7.000000

Name: Full_meals_eaten, dtype: float64

The number of full meals eaten while in the hospital can indicate how well a patient is recovering. If a patient is not eating that could mean they don't feel well or aren't physically able to eat. The mean is 1.00 and the median is also 1.00. 75% or less of patients ate 2 meals which indicates that most patients were not eating full meals.

HighBlood: Whether the patient has high blood pressure (yes, no)

Summary Statistics for HighBlood :

count	10000
unique	2
top	No
freq	5910

Name: HighBlood, dtype: object

Approximately 59% of patients reported had high blood pressure, while the other almost 41% reported not having high blood pressure. Summary statistics provide information that "no" was the top value indicating that most people did not have High Blood Pressure.

Stroke: Whether the patient has had a stroke (yes, no)

Summary Statistics for Stroke :

Count	10000
unique	2

top No
freq 8007
Name: Stroke, dtype: object

For stroke, we see about 20% of patients have had a stroke and approximately 80% have not. Summary statistics show that "no" is the top answer out of two answers, meaning most patients had not had a stroke.

Overweight: Whether the patient is considered overweight based on age, gender, and height (yes, no)

Summary Statistics for Overweight :
count 10000
unique 2
top Yes
freq 7094
Name: Overweight, dtype: object

Patients age, gender, height, and weight make up the category of overweight. Based on these factors, the top answer is "yes" with 70.94% being considered overweight.

Arthritis: Whether the patient has arthritis (yes, no)

Summary Statistics for Arthritis :
count 10000
unique 2
top No
freq 6426
Name: Arthritis, dtype: object

A patient either has arthritis or they do not, based on those choices the top answer is "no" with 64.26% of patients not being listed as having arthritis.

Diabetes: Whether the patient has diabetes (yes, no)

Summary Statistics for Diabetes :
count 10000
unique 2
top No
freq 7262
Name: Diabetes, dtype: object

A patient either has diabetes or they do not, based on those choices the top answer is "no" with 72.62% of patients not being listed as having diabetes.

Hyperlipidemia: Whether the patient has hyperlipidemia (yes, no)

Summary Statistics for Hyperlipidemia :
count 10000
unique 2
top No
freq 6628
Name: Hyperlipidemia, dtype: object

There are two options, yes/no, for if a patient has hyperlipidemia. 66.28% of patients are listed as not having hyperlipidemia.

BackPain: Whether the patient has chronic back pain (yes, no)

Summary Statistics for BackPain :
count 10000
unique 2

top No
freq 5886
Name: BackPain, dtype: float64

Chronic back pain symptoms were not prevalent in 58.86% of patients based on the above statistics.

Anxiety: Whether the patient has an anxiety disorder (yes, no)

Summary Statistics for Anxiety :

count 10000
unique 2
top No
freq 6785
Name: Anxiety, dtype: float64

Patients listed as having an anxiety disorder were not the norm in the data provided. 67.85% of patients are listed as not having an anxiety disorder.

Asthma: Whether the patient has asthma (yes, no)

Summary Statistics for Asthma :

count 10000
unique 2
top No
freq 7107
Name: Asthma, dtype: float64

Patients listed as having asthma were not the norm in the data provided. 71.07% of patients are listed as not having asthma.

Initial_days: The number of days the patient stayed in the hospital during the initial visit

Summary Statistics for Initial_days :

count 10000.000000
mean 33.956000
std 26.301628
min 1.000000
25% 7.000000
50% 35.500000
75% 61.000000
max 71.000000
Name: Initial_days, dtype: float64

Initial days spent in the hospital during a patient's initial stay ranged from 1 day to 71 days with the mean being 33.95 days and the median being 35.50 days.

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

Summary Statistics for TotalCharge :

count 10000.000000
mean 5311.673500
std 2180.391406
min 1938.000000
25% 3179.000000
50% 5213.500000
75% 7459.250000
max 9180.000000
Name: TotalCharge, dtype: float64

The mean total charges for a patient came to be 5,311.67 while the median came out at 5,213.50. The minimum total charges were 1,938.00 while the maximum charges were 9,180.00.

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

Summary Statistics for Additional_charges :

count	10000.000000
mean	12934.032300
std	6542.600277
min	3125.000000
25%	7985.750000
50%	11573.500000
75%	15626.000000
max	30566.000000

Name: Additional_charges, dtype: float64

Additional charges has a high standard deviation at 6542.60 with a minimum of 3125.70 additional charges and 30,566.07 as the max additional charges. The mean additional charges value is 12934.52 and the median is 11573.97 so they are close to symmetrical distributed.

Initial_admin: The means by which the patient was admitted into the hospital initially (emergency a dmission, elective admission, observation)

Summary Statistics for Initial_admin :

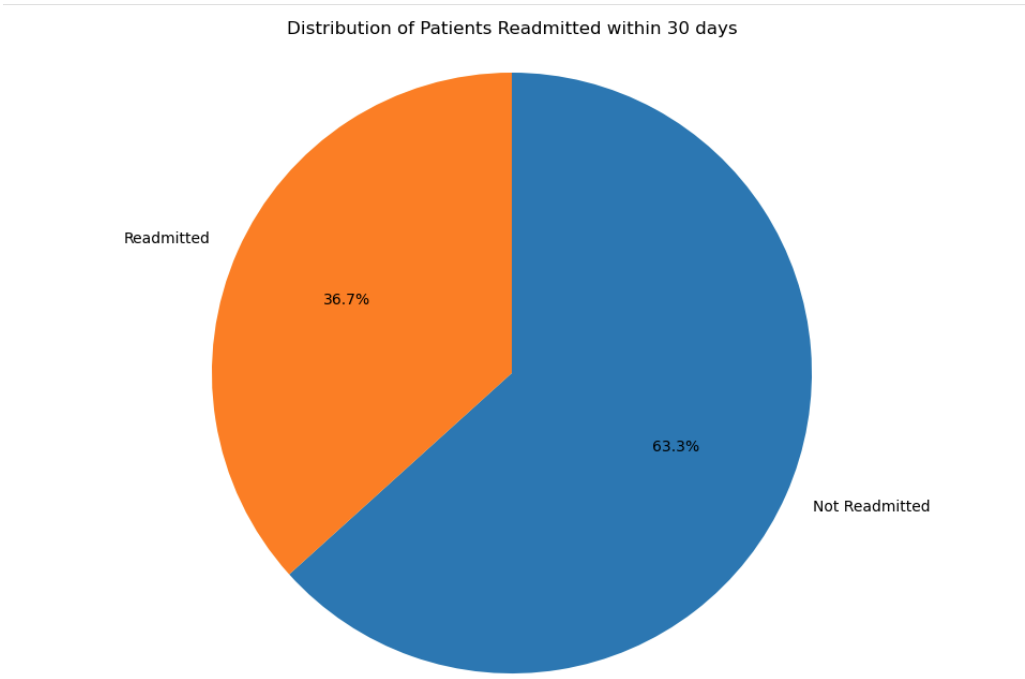
count	10000
unique	3
top	Emergency Admission
freq	5060

Name: Initial_admin, dtype: object

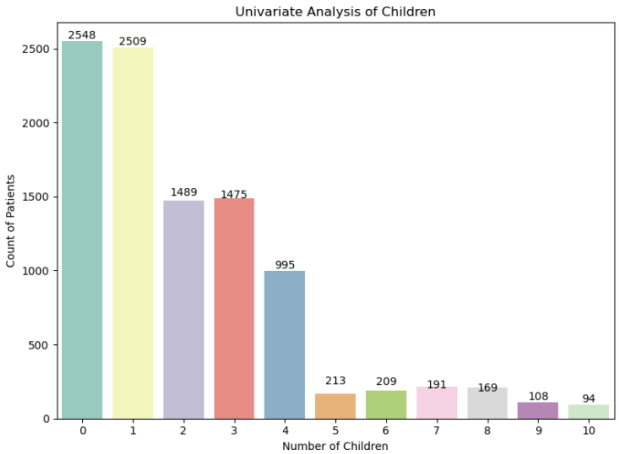
There are three categories of admission for patients. Approximately 50% are admitted via emergency procedures with another 25% being admitted for elective procedures and almost 25% for observation. The top value, according to the summary statistics, is Emergency Admission.

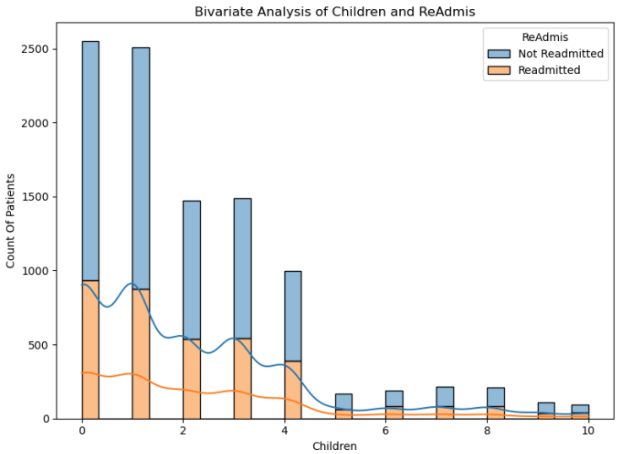
3. Generate univariate and bivariate visualizations of the distributions of the dependent and independent variables, including the dependent variable in your bivariate visualizations.

```
# Dependent Variable Visualization ReAdmis
# Piechart for ReAdmis
plt.figure(figsize=[12, 8])
plt.title('Distribution of Patients Readmitted within 30 days')
ReAdmis_counts = df.ReAdmis.value_counts()
ReAdmis_labels = ["Not Readmitted", "Readmitted"]
plt.pie(ReAdmis_counts, labels=ReAdmis_labels, autopct='%1.1f%%', startangle=90,
counterclock=False)
plt.axis('equal')
plt.show()
```

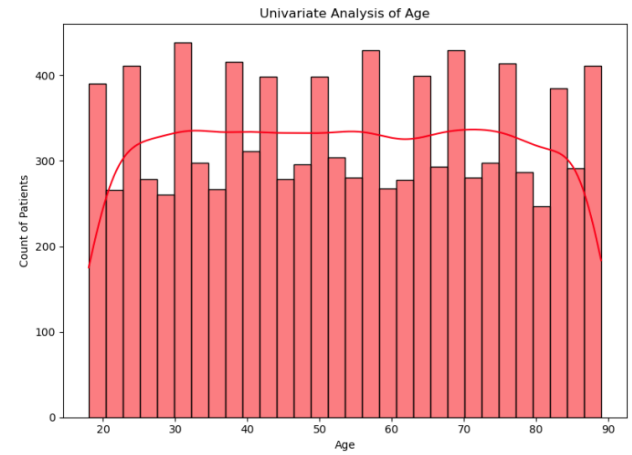


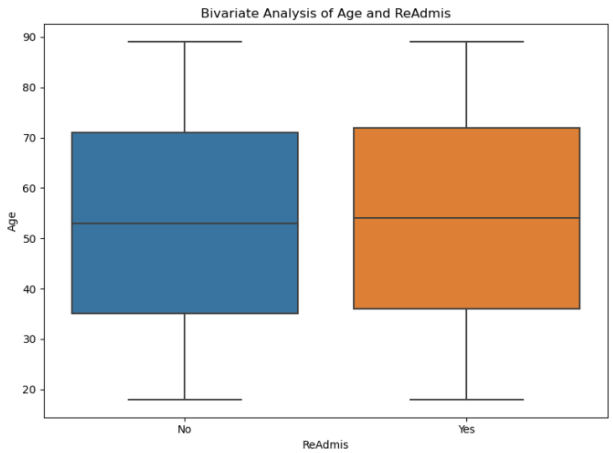

```
# Children Univariate and Bivariate
Children_counts = df["Children"].value_counts()
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Bar chart for Children
sns.barplot(x=Children_counts.index, y=Children_counts.values, palette="Set3", ax=ax1)
ax1.set_title('Univariate Analysis of Children')
ax1.set_xlabel('Number of Children')
ax1.set_ylabel('Count of Patients')
for index, value in enumerate(Children_counts.values):
    ax1.text(index, value, str(value), ha='center', va='bottom')

# Histogram for Children and ReAdmis
sns.histplot(data=df, x="Children", hue="ReAdmis", bins=30, kde=True, multiple="stack", ax=ax2)
ax2.set_title("Bivariate Analysis of Children and ReAdmis")
ax2.set_xlabel("Children")
ax2.set_ylabel("Count Of Patients")
handles, labels = ax2.get_legend().legend_handles, ["Not Readmitted", "Readmitted"]
ax2.legend(handles=handles, labels=labels, title="ReAdmis")

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





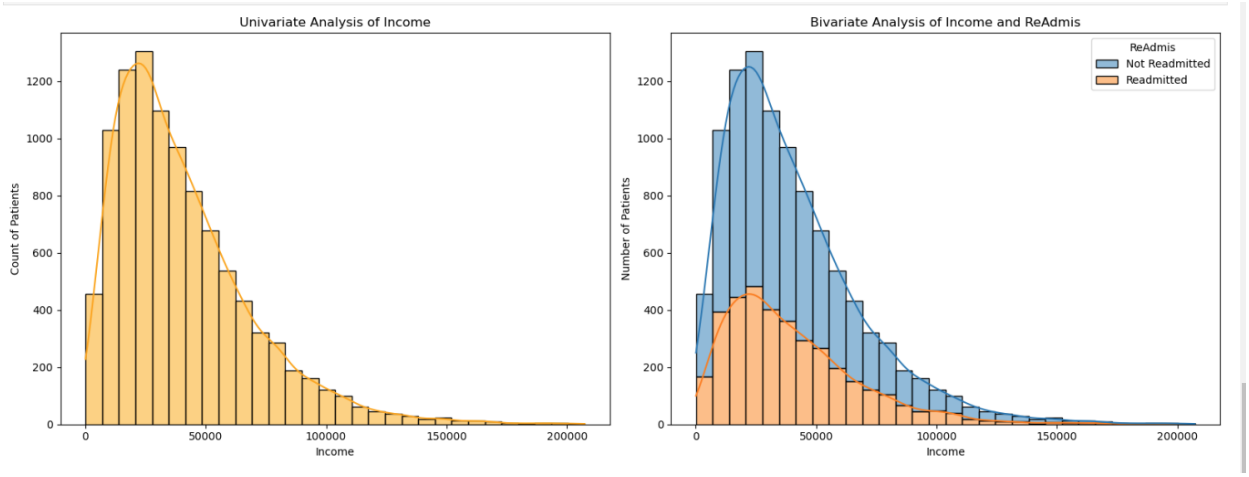

```
# Age Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

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

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

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



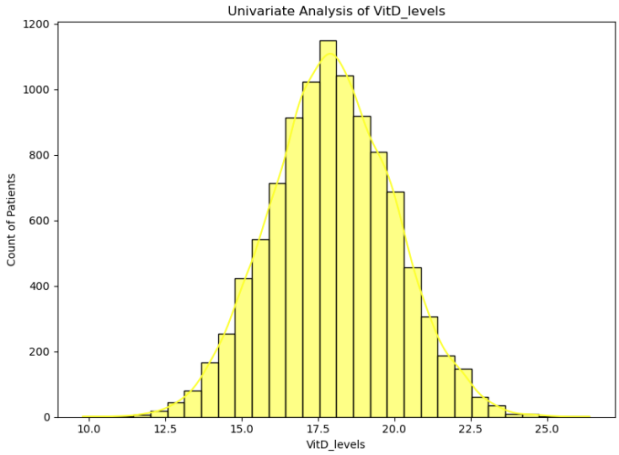


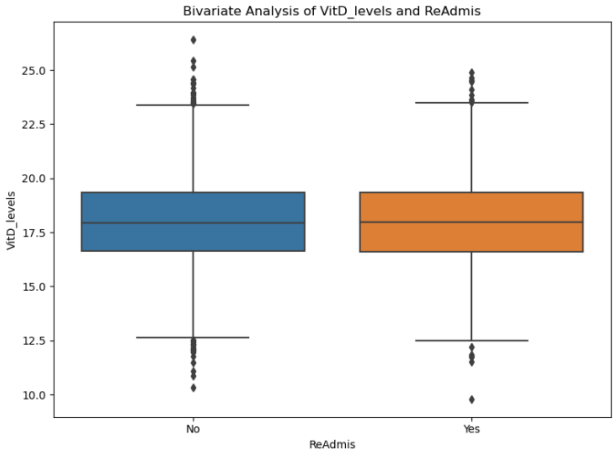

```
# Income Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Histogram for Income
sns.histplot(df["Income"], kde=True, bins=30, color="orange", ax=ax1)
```

```
ax1.set_title('Univariate Analysis of Income')
ax1.set_xlabel('Income')
ax1.set_ylabel('Count of Patients')

# Histogram for Income and ReAdmis
sns.histplot(data=df, x="Income", hue="ReAdmis", bins=30, kde=True, multiple="stack", ax=ax2)
ax2.set_title("Bivariate Analysis of Income and ReAdmis")
ax2.set_xlabel("Income")
ax2.set_ylabel("Number of Patients")
handles, labels = ax2.get_legend().legend_handles, ["Not Readmitted", "Readmitted"]
ax2.legend(handles=handles, labels=labels, title="ReAdmis")

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

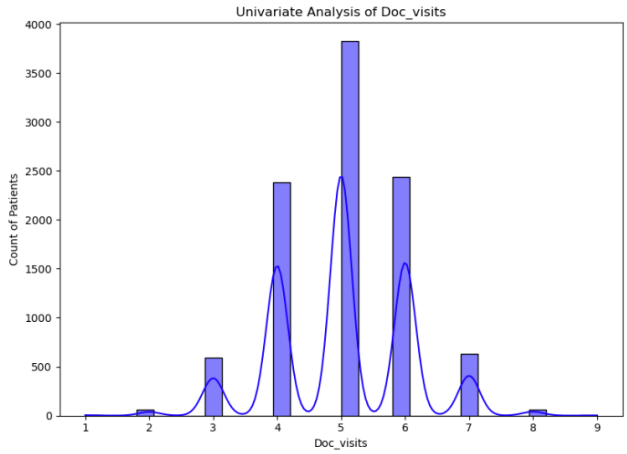


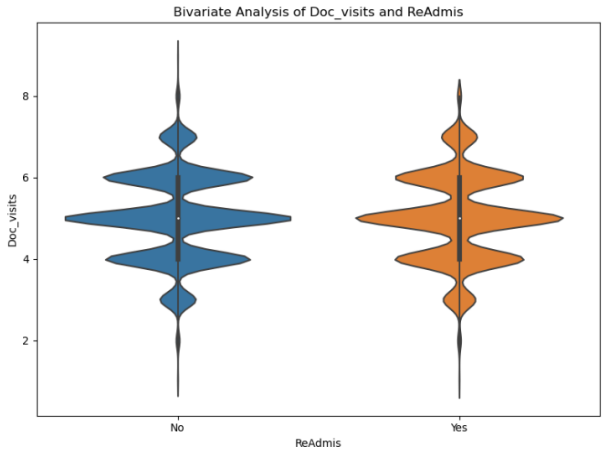

```
# VitD_levels Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

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

# Boxplot for VitD_levels and ReAdmis
sns.boxplot(x=df["ReAdmis"], y=df["VitD_levels"], ax=ax2)
ax2.set_title('Bivariate Analysis of VitD_levels and ReAdmis')
ax2.set_xlabel('ReAdmis')
ax2.set_ylabel('VitD_levels')

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





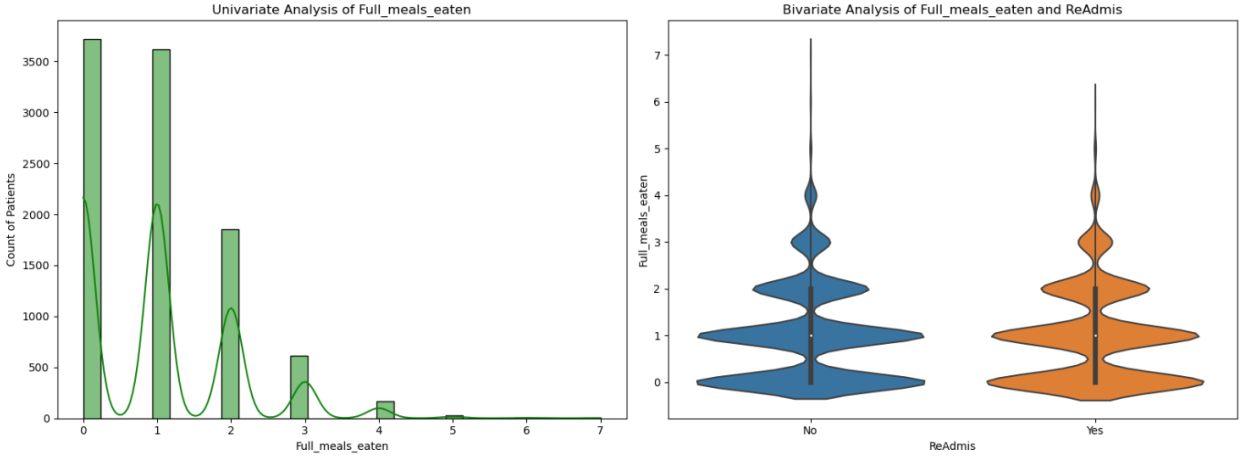

```
# Doc_visits Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

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

# Violin plot for Doc_visits and ReAdmis
sns.violinplot(x=df["ReAdmis"], y=df["Doc_visits"], ax=ax2)
ax2.set_title('Bivariate Analysis of Doc_visits and ReAdmis')
ax2.set_xlabel('ReAdmis')
ax2.set_ylabel('Doc_visits')

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





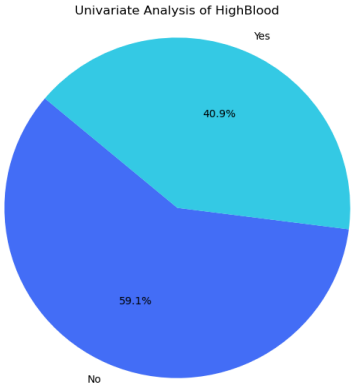

```
# Full_meals_eaten Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))
```

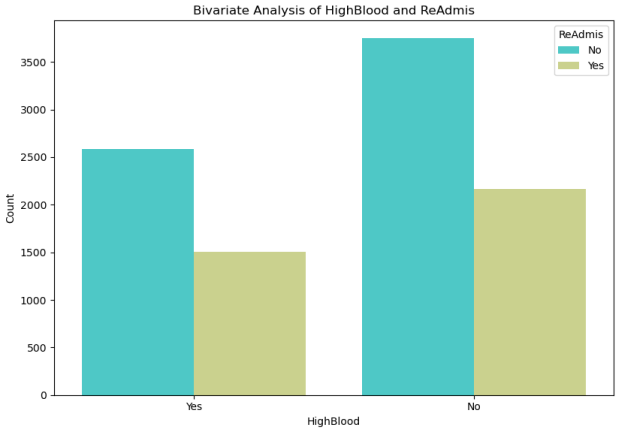
```
# Histogram for Full_meals_eaten
sns.histplot(df["Full_meals_eaten"], kde=True, bins=30, color="green", ax=ax1)
ax1.set_title('Univariate Analysis of Full_meals_eaten')
ax1.set_xlabel('Full_meals_eaten')
ax1.set_ylabel('Count of Patients')

# Violin plot for Full_meals_eaten and ReAdmis
sns.violinplot(x=df["ReAdmis"], y=df["Full_meals_eaten"], ax=ax2)
ax2.set_title('Bivariate Analysis of Full_meals_eaten and ReAdmis')
ax2.set_xlabel('ReAdmis')
ax2.set_ylabel('Full_meals_eaten')

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

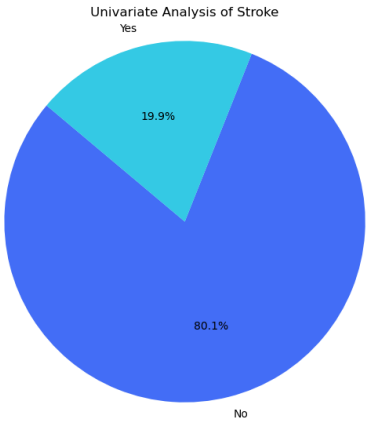


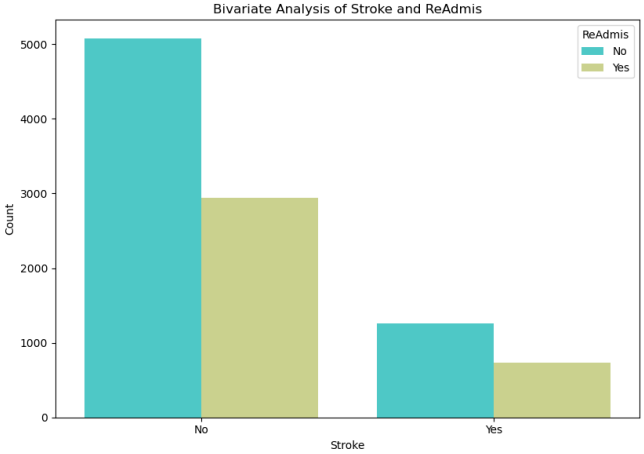

```
# Generate visualizations yes/no variables
variables = ["HighBlood", "Stroke", "Overweight", "Arthritis", "Diabetes", "Hyperlipidemia",
"BackPain", "Anxiety", "Asthma"]

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

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

    # Count Plot (Bivariate Analysis)
    sns.countplot(x=df[var], hue=df["ReAdmis"], palette="rainbow", ax=ax2)
    ax2.set_title(f'Bivariate Analysis of {var} and ReAdmis')
    ax2.set_xlabel(var)
    ax2.set_ylabel('Count')

    plt.tight_layout()
```

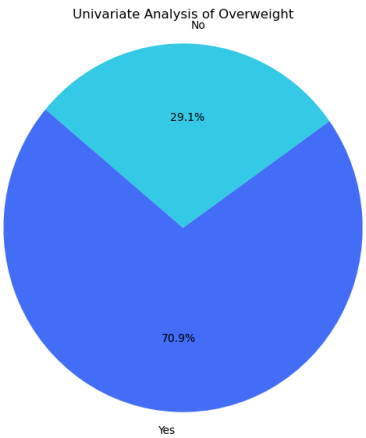
plt.show()

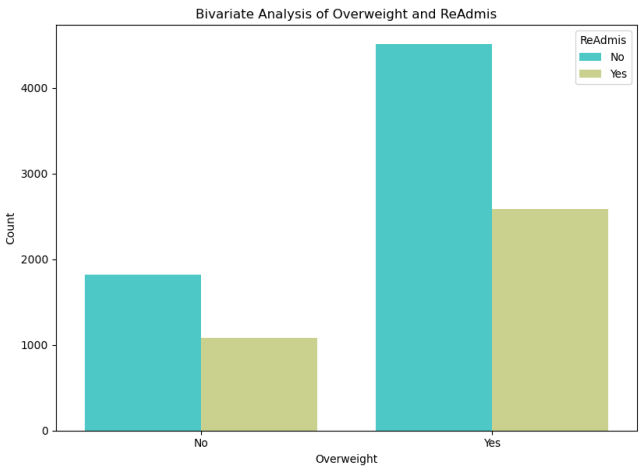

Univariate Analysis of HighBlood
Yes
40.9%
59.1%
No

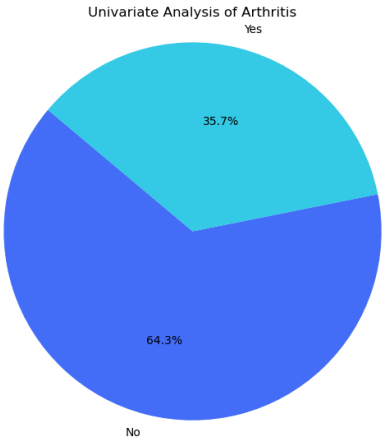

Bivariate Analysis of HighBlood and ReAdmis
ReAdmis
No
Yes
Count
HighBlood
Yes
No

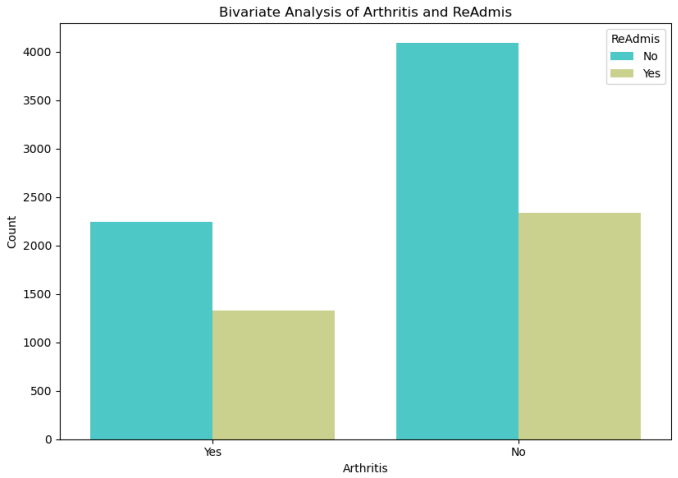

Univariate Analysis of Stroke
Yes
19.9%
80.1%
No

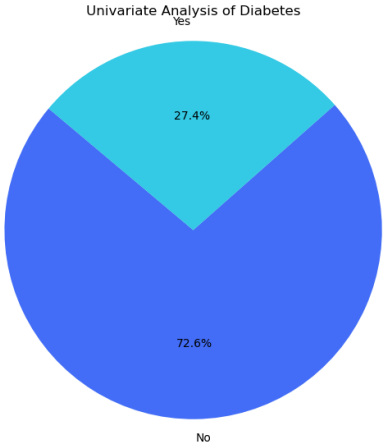

Bivariate Analysis of Stroke and ReAdmis
ReAdmis
No
Yes
Count
Stroke
No
Yes

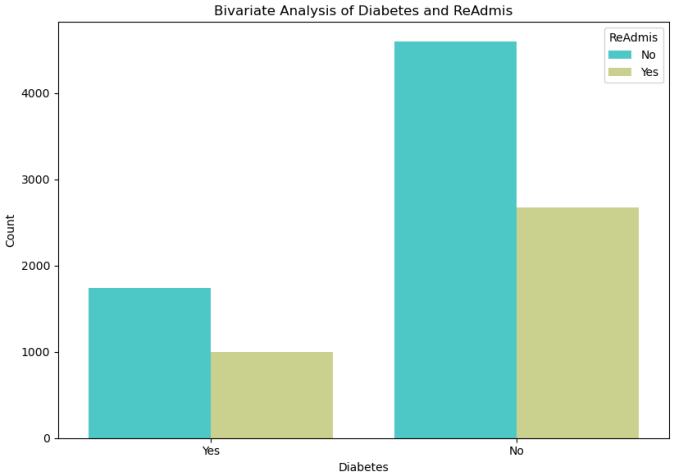

Univariate Analysis of Overweight
No
29.1%
70.9%
Yes

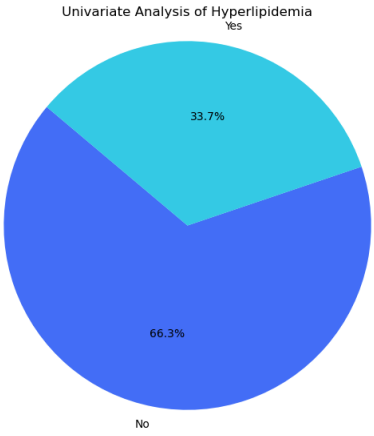

Bivariate Analysis of Overweight and ReAdmis
ReAdmis
No
Yes
Count
Overweight
No
Yes

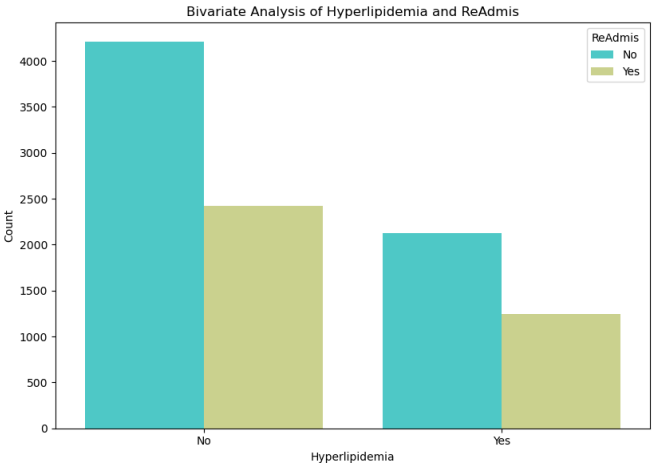

Univariate Analysis of Arthritis
Yes
35.7%
64.3%
No

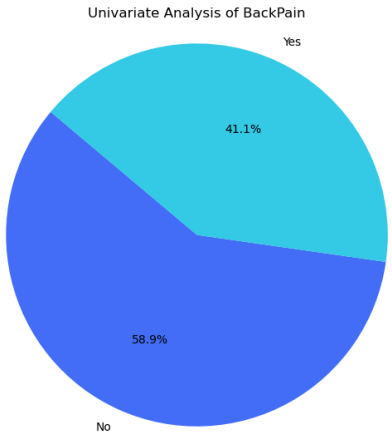

Bivariate Analysis of Arthritis and ReAdmis
ReAdmis
No
Yes
Count
0
500
1000
1500
2000
2500
3000
3500
4000
Yes
No
Arthritis

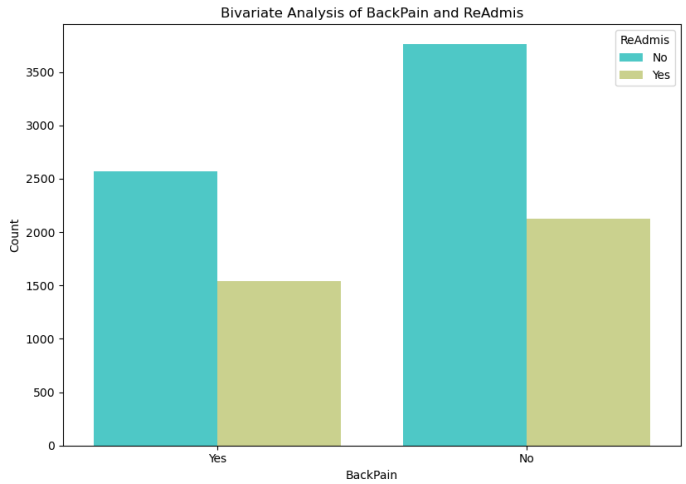

Univariate Analysis of Diabetes
Yes
27.4%
72.6%
No

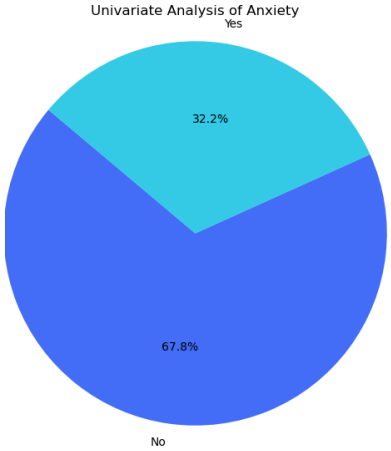

Bivariate Analysis of Diabetes and ReAdmis
ReAdmis
No
Yes
Count
0
1000
2000
3000
4000
Yes
No
Diabetes

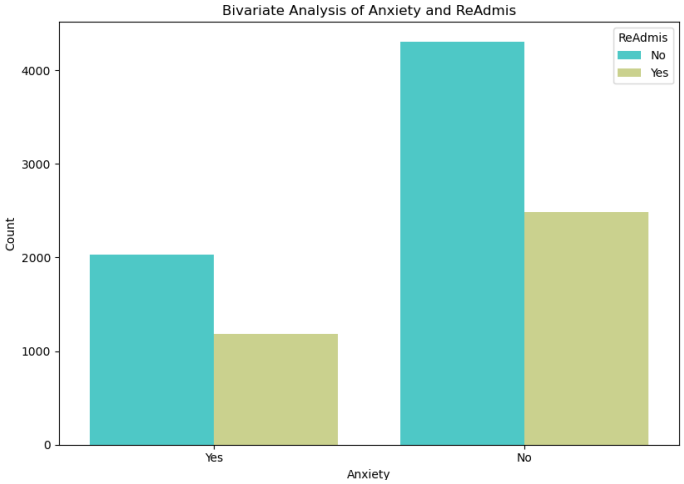

Univariate Analysis of Hyperlipidemia
Yes
33.7%
66.3%
No

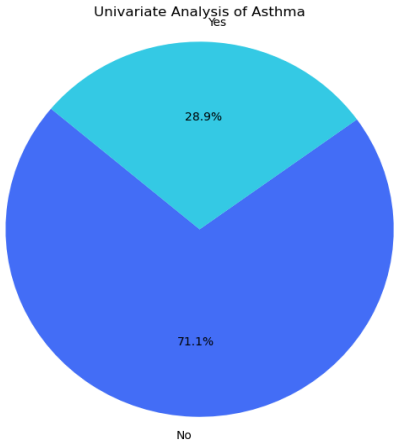

Bivariate Analysis of Hyperlipidemia and ReAdmis
ReAdmis
No
Yes
Count
0
500
1000
1500
2000
2500
3000
3500
4000
No
Yes
Hyperlipidemia

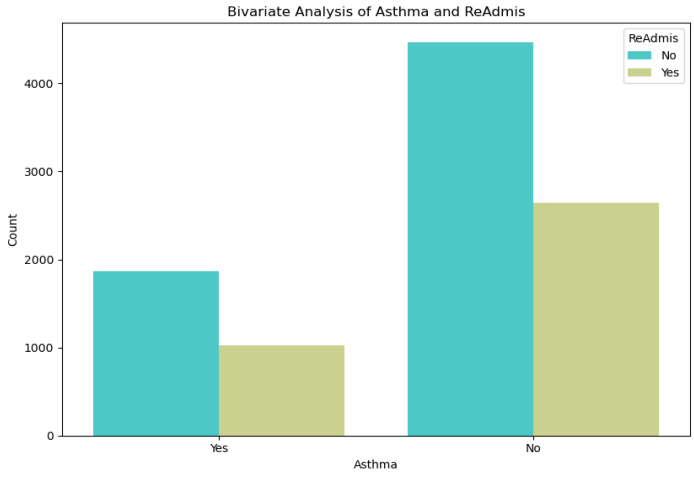






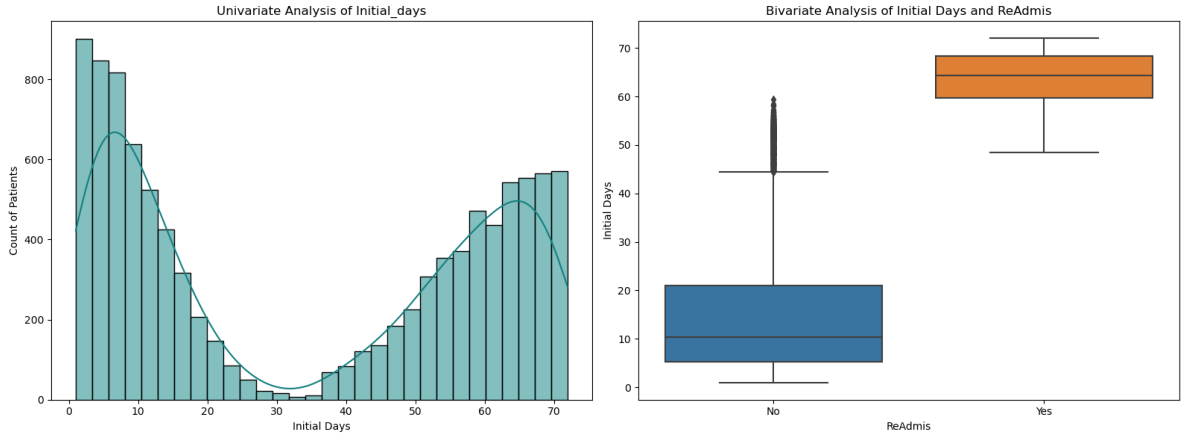




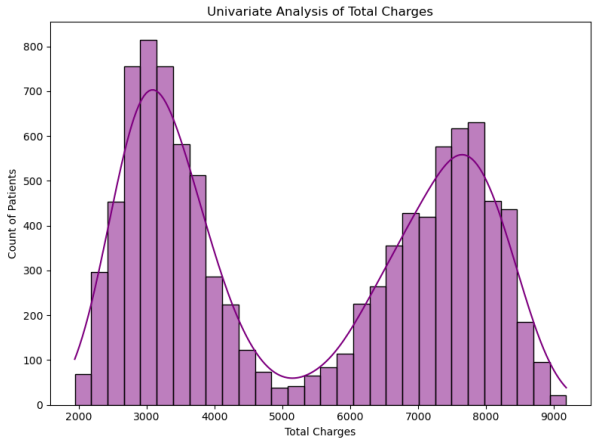


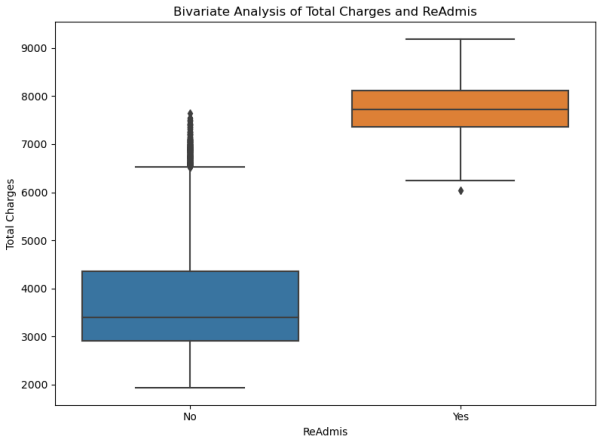

```
# Initial_days Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Histogram for Initial_days
sns.histplot(df["Initial_days"], kde=True, bins=30, color='teal', ax=ax1)
ax1.set_title('Univariate Analysis of Initial_days')
```

```
ax1.set_xlabel('Initial Days')
ax1.set_ylabel('Count of Patients')

# Box Plot for Initial_days and ReAdmis
sns.boxplot(x=df["ReAdmis"], y=df["Initial_days"], ax=ax2)
ax2.set_title('Bivariate Analysis of Initial Days and ReAdmis')
ax2.set_xlabel('ReAdmis')
ax2.set_ylabel('Initial Days')

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

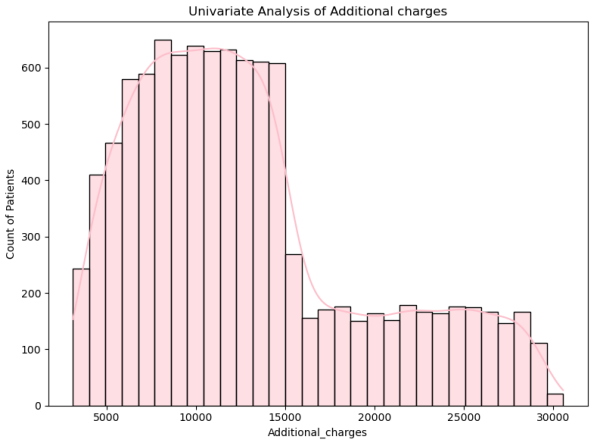


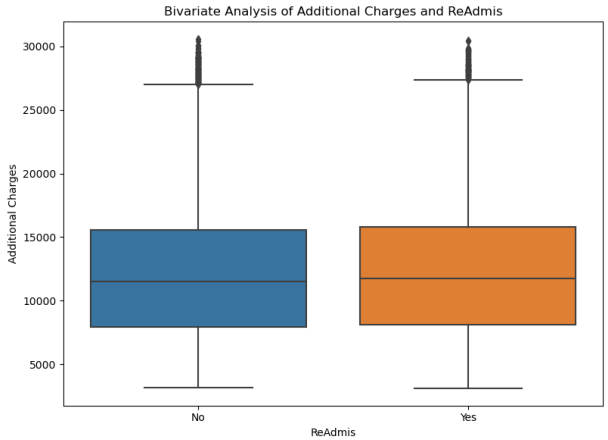

```
# TotalCharge Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

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

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

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





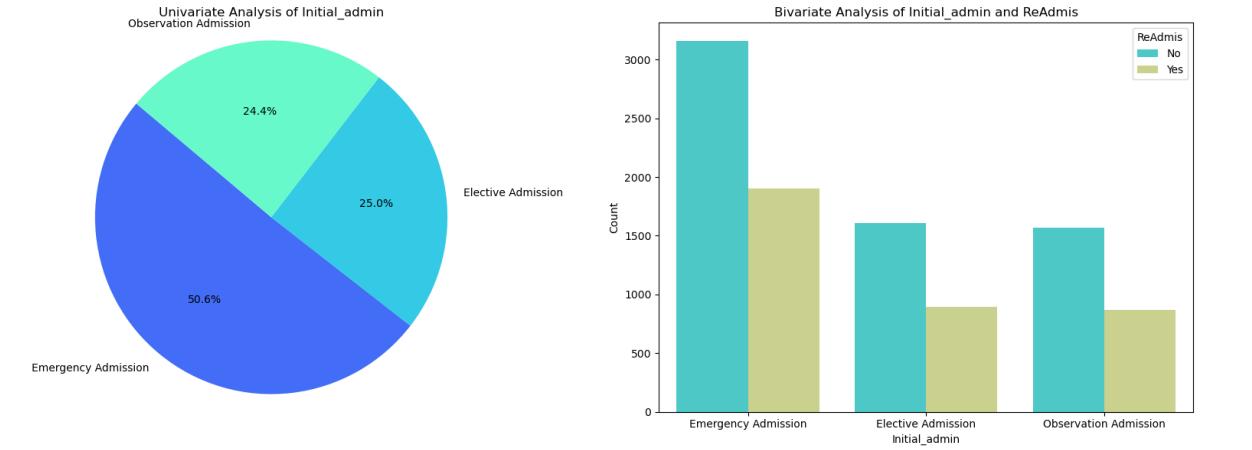

```
# Additional_charges Univariate and Bivariate
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

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

# Box Plot for Additional_charges and ReAdmis
sns.boxplot(x=df["ReAdmis"], y=df["Additional_charges"], ax=ax2)
ax2.set_title('Bivariate Analysis of Additional Charges and ReAdmis')
ax2.set_xlabel('ReAdmis')
ax2.set_ylabel('Additional Charges')

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






```
# Initial_admin Univariate and Bivariate
Initial_admin_counts = df["Initial_admin"].value_counts()

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

```
# Pie Chart for Initial_admin
ax1.pie(Initial_admin_counts, labels=Initial_admin_counts.index, autopct='%1.1f%%',
startangle=140, colors=sns.color_palette("rainbow"))
ax1.set_title('Univariate Analysis of Initial_admin')
ax1.axis('equal')

# Count Plot for Initial_admin and ReAdmis
sns.countplot(x=df["Initial_admin"], hue=df["ReAdmis"], palette="rainbow", ax=ax2)
ax2.set_title('Bivariate Analysis of Initial_admin and ReAdmis')
ax2.set_xlabel('Initial_admin')
ax2.set_ylabel('Count')

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



4. Describe your data transformation goals that align with your research question and the steps used to transform the data to achieve the goals, including the annotated code.
   The goals of data transformation for this analysis are to align needed variables to the research method after determining which variables would be used.
   First the data needed to be inspected, to do that a display data types and visual inspection of the first file rows of data were called via coding.

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

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

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

   Variables that were of type 'float64' were aligned to astype(int). These variables were "Income", "Initial_days", "TotalCharge", and "Additional_charges".

```
# Update currency to 3 decimal places
df["Income"] = df["Income"].astype(int)
```

```
# Update Initial days to 3 decimal places
df["Initial_days"] = df["Initial_days"].astype(int)

# Update Total Charges to 3 decimal places
df["TotalCharge"] = df["TotalCharge"].astype(int)

# Update Additional Charges to 3 decimal places
df["Additional_charges"] = df["Additional_charges"].astype(int)
```

The variables that were of “object” type needed to be converted to “category” or “bool”.

```
# Convert columns to boolean
bool_mapping = {"Yes": 1, "No": 0}
columns_to_convert = ["HighBlood", "Stroke", "Overweight", "Arthritis", "Diabetes",
"Hyperlipidemia", "BackPain", "Anxiety", "Asthma", "ReAdmis"]
for col in columns_to_convert:
    df[col] = df[col].map(bool_mapping)
```

The categorical column of “Initial_admin” was adjusted with one hot encoding as explained by Dr. Middleton (Western Governors University, 2022) in the course videos.

```
# Generate columns of dummy values
initial_admit_df = pd.get_dummies(data=df["Initial_admin"], drop_first=True)
```

The new data frame was then created with the model variables as follows:

```
# Create new df with model variables
LogReg_df = df[["Children", "Age", "Income", "Doc_visits", "Full_meals_eaten", "HighBlood",
"Stroke", "Overweight", "Arthritis", "Diabetes", "Hyperlipidemia", "BackPain", "Anxiety", "Asthma",
"Initial_days", "TotalCharge", "Additional_charges", "ReAdmis"]].copy()

# Dummies for Initial Admit
LogReg_df["initial_admit_elect"] = initial_admit_df["Emergency Admission"].astype(int)
LogReg_df["initial_admit_obs"] = initial_admit_df["Observation Admission"].astype(int)
```

5. Provide the prepared data set as a CSV file.

```
# Save LogReg_df to a CSV file
LogReg_df.to_csv("LogReg_df.csv", index=False)
print("LogReg_df has been saved to LogReg_df.csv'.")
```

Part IV: Model Comparison and Analysis

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

1. The initial logistic regression model coding is listed below:

```
y = LogReg_df.ReAdmis
X = LogReg_df[["Children", "Age", "Income", "Doc_visits", "Full_meals_eaten", "HighBlood",
"Stroke", "Overweight", "Arthritis", "Diabetes", "Hyperlipidemia", "BackPain", "Anxiety", "Asthma",
"Initial_days", "TotalCharge", "Additional_charges", "initial_admit_obs", "initial_admit_elect"
]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.038679
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                 ReAdmis   No. Observations:                10000
Model:                           Logit   Df Residuals:                     9980
Method:                            MLE   Df Model:                           19
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                  0.9412
Time:                        15:00:46   Log-Likelihood:                -386.79
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
Children              0.0693      0.042      1.663      0.096      -0.012       0.151
Age                  -0.0183      0.014     -1.346      0.178      -0.045       0.008
Income             1.444e-06   3.31e-06      0.436      0.663   -5.05e-06    7.94e-06
Doc_visits           -0.0470      0.087     -0.539      0.590      -0.218       0.124
Full_meals_eaten      0.0684      0.095      0.720      0.472      -0.118       0.254
HighBlood            -0.1667      0.535     -0.312      0.755      -1.216       0.882
Stroke                1.4240      0.246      5.784      0.000       0.941       1.906
Overweight           -0.1773      0.208     -0.852      0.394      -0.585       0.230
Arthritis            -1.2233      0.209     -5.862      0.000      -1.632      -0.814
Diabetes              0.3064      0.213      1.439      0.150      -0.111       0.724
Hyperlipidemia        0.1579      0.205      0.772      0.440      -0.243       0.559
BackPain              0.1213      0.192      0.631      0.528      -0.256       0.499
Anxiety              -0.9264      0.206     -4.492      0.000      -1.331      -0.522
Asthma               -1.1664      0.212     -5.507      0.000      -1.581      -0.751
Initial_days          1.1305      0.070     16.259      0.000       0.994       1.267
TotalCharge           0.0016      0.000      3.375      0.001       0.001       0.003
Additional_charges 8.935e-05   5.78e-05      1.545      0.122    -2.4e-05       0.000
initial_admit_obs     0.6766      0.260      2.602      0.009       0.167       1.186
initial_admit_elect   1.2038      0.332      3.625      0.000       0.553       1.855
const               -72.1225      3.850    -18.735      0.000     -79.668     -64.577
=====================================================================================

Possibly complete quasi-separation: A fraction 0.79 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

2. In order to determine a statistically based model evaluation metric to reduce the initial model, a variance inflation factor (VIF) was completed using statsmodels. Statsmodels is a Python module that easily allows various statistical models to be explored (Sharma, 2020). Using this method also allows us to see if any variables should be eliminated due to high multicollinearity.

3. Provide a reduced logistic regression model that follows the feature selection or model evaluation process in part D2, including a screenshot of the output for each model.

```
# Check for VIF to see if variables should be eliminated due to high multicolinearity
X = LogReg_df[["Children", "Age", "Income", "Doc_visits", "Full_meals_eaten", "HighBlood",
"Stroke", "Overweight", "Arthritis", "Diabetes", "Hyperlipidemia", "BackPain", "Anxiety", "Asthma",
"Initial_days", "TotalCharge", "Additional_charges", "initial_admit_obs", "initial_admit_elect" ]]

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

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

print(VIF_df)
```

	feature	variance_inflation_factor
0	Children	1.933189
1	Age	65.238489
2	Income	2.986620
3	Doc_visits	20.946799
4	Full_meals_eaten	1.981936
5	HighBlood	12.770898
6	Stroke	1.255289
7	Overweight	3.387159
8	Arthritis	1.588368
9	Diabetes	1.404547
10	Hyperlipidemia	1.562255
11	BackPain	1.751698
12	Anxiety	1.514600
13	Asthma	1.404350
14	Initial_days	100.456895
15	TotalCharge	247.175370
16	Additional_charges	75.079465
17	initial_admit_obs	1.947861
18	initial_admit_elect	4.679702

Based on the above VIF values, we see that Age, Doc_Visits, HighBlood, Initial_days, TotalCharges, and Additional_charges have values over 10. Generally any value over 10 indicates significant multicollinearity. Additional recalculation and reduction need to occur.

```
# Remove the variable with the highest VIF -Round 1
X_reduced = X.drop(columns=["TotalCharge"])

VIF_df_reduced = pd.DataFrame()
VIF_df_reduced["feature"] = X_reduced.columns
VIF_df_reduced["variance_inflation_factor"] = [variance_inflation_factor(X_reduced.values, i) for i
in range(len(X_reduced.columns))]
```

print(VIF_df_reduced)

	feature	variance_inflation_factor
0	Children	1.908069
1	Age	61.047681
2	Income	2.891563
3	Doc_visits	12.949081
4	Full_meals_eaten	1.948696
5	HighBlood	12.089586
6	Stroke	1.248589
7	Overweight	3.268955
8	Arthritis	1.544814
9	Diabetes	1.370538
10	Hyperlipidemia	1.489488
11	BackPain	1.686936
12	Anxiety	1.462477
13	Asthma	1.396181
14	Initial_days	2.580481
15	Additional_charges	73.290285
16	initial_admit_obs	1.911809
17	initial_admit_elect	2.860328

```
# Remove the variable with the second highest VIF – Round 2
X_reduced = X_reduced.drop(columns=["Additional_charges"])

# Recalculate VIFs for the reduced set of variables
VIF_df_reduced = pd.DataFrame()
VIF_df_reduced["feature"] = X_reduced.columns
VIF_df_reduced["variance_inflation_factor"] = [variance_inflation_factor(X_reduced.values, i) for i
in range(len(X_reduced.columns))]

print(VIF_df_reduced)
```

	feature	variance_inflation_factor
0	Children	1.907552
1	Age	6.638593
2	Income	2.883181
3	Doc_visits	12.059400
4	Full_meals_eaten	1.947189
5	HighBlood	1.675633
6	Stroke	1.243268
7	Overweight	3.259674
8	Arthritis	1.541610
9	Diabetes	1.370473
10	Hyperlipidemia	1.488112
11	BackPain	1.684542
12	Anxiety	1.462024
13	Asthma	1.395971
14	Initial_days	2.570796
15	initial_admit_obs	1.901441
16	initial_admit_elect	2.853043

```
# Remove the variable with the high VIF – Round 3
X_reduced = X_reduced.drop(columns=["Doc_visits"])

# Recalculate VIFs for the reduced set of variables
VIF_df_reduced = pd.DataFrame()
VIF_df_reduced["feature"] = X_reduced.columns
VIF_df_reduced["variance_inflation_factor"] = [variance_inflation_factor(X_reduced.values, i) for i
in range(len(X_reduced.columns))]

print(VIF_df_reduced)
```

	feature	variance_inflation_factor
0	Children	1.874268
1	Age	5.484477
2	Income	2.719039
3	Full_meals_eaten	1.906837
4	HighBlood	1.652404
5	Stroke	1.236977
6	Overweight	3.043625
7	Arthritis	1.524935
8	Diabetes	1.356514
9	Hyperlipidemia	1.476875
10	BackPain	1.661943
11	Anxiety	1.449076
12	Asthma	1.388774
13	Initial_days	2.478382
14	initial_admit_obs	1.803355
15	initial_admit_elect	2.647751

After addressing the multicollinearity issue, the method that best fits is to use backwards elimination. The stepwise or backward elimination method involves starting with all predictor variables and iteratively removing the least significant variable based on the highest p-value, until all variables in

the model have a p-value below a specific threshold(Neter et al., 1996). The threshold here will be p<=0.05. Anything above that will be removed and whatever variables remain will be discussed.

```
#  Perform backward elimination - recheck after VIF reduction
y = LogReg_df.ReAdmis
X = LogReg_df[["Children", "Age", "Income", "Full_meals_eaten", "HighBlood", "Stroke",
"Overweight", "Arthritis", "Diabetes", "Hyperlipidemia", "BackPain", "Anxiety", "Asthma",
"Initial_days", "initial_admit_obs", "initial_admit_elect" ]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.039453
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9983
Method:                           MLE   Df Model:                           16
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                  0.9400
Time:                        15:32:54   Log-Likelihood:                -394.53
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==========================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
Children                   0.0776      0.041      1.870      0.061      -0.004       0.159
Age                        0.0017      0.004      0.380      0.704      -0.007       0.010
Income                  1.444e-06   3.29e-06      0.439      0.660      -5e-06    7.88e-06
Full_meals_eaten           0.0823      0.093      0.882      0.378      -0.101       0.265
HighBlood                  0.7684      0.194      3.957      0.000       0.388       1.149
Stroke                     1.4559      0.241      6.044      0.000       0.984       1.928
Overweight                -0.1438      0.205     -0.702      0.483      -0.545       0.258
Arthritis                 -1.0904      0.201     -5.432      0.000      -1.484      -0.697
Diabetes                   0.4218      0.208      2.031      0.042       0.015       0.829
Hyperlipidemia             0.3471      0.196      1.772      0.076      -0.037       0.731
BackPain                   0.2491      0.187      1.335      0.182      -0.117       0.615
Anxiety                   -0.8257      0.201     -4.115      0.000      -1.219      -0.432
Asthma                    -1.1480      0.210     -5.464      0.000      -1.560      -0.736
Initial_days               1.2398      0.063     19.632      0.000       1.116       1.364
initial_admit_obs          0.6266      0.257      2.438      0.015       0.123       1.130
initial_admit_elect        2.0363      0.243      8.393      0.000       1.561       2.512
const                    -68.0894      3.473    -19.608      0.000     -74.896     -61.283
==========================================================================================

Possibly complete quasi-separation: A fraction 0.78 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

```
#  Perform backward elimination #1 remove "Age" p=0.704
y = LogReg_df.ReAdmis
```

```
X = LogReg_df[["Children", "Income", "Full_meals_eaten", "HighBlood", "Stroke", "Overweight",
"Arthritis", "Diabetes", "Hyperlipidemia", "BackPain", "Anxiety", "Asthma", "Initial_days",
"initial_admit_obs", "initial_admit_elect" ]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.039460
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                 ReAdmis   No. Observations:                10000
Model:                           Logit   Df Residuals:                     9984
Method:                            MLE   Df Model:                           15
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                  0.9400
Time:                        15:34:20   Log-Likelihood:                -394.60
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==========================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
Children                   0.0777      0.042      1.870      0.061      -0.004       0.159
Income                1.398e-06   3.28e-06      0.426      0.670   -5.04e-06    7.83e-06
Full_meals_eaten           0.0822      0.093      0.881      0.378      -0.101       0.265
HighBlood                  0.7699      0.194      3.967      0.000       0.389       1.150
Stroke                     1.4563      0.241      6.044      0.000       0.984       1.929
Overweight                -0.1429      0.205     -0.697      0.486      -0.544       0.259
Arthritis                 -1.0890      0.201     -5.428      0.000      -1.482      -0.696
Diabetes                   0.4159      0.207      2.009      0.044       0.010       0.822
Hyperlipidemia             0.3481      0.196      1.777      0.076      -0.036       0.732
BackPain                   0.2486      0.187      1.333      0.183      -0.117       0.614
Anxiety                   -0.8226      0.200     -4.103      0.000      -1.215      -0.430
Asthma                    -1.1478      0.210     -5.461      0.000      -1.560      -0.736
Initial_days               1.2405      0.063     19.637      0.000       1.117       1.364
initial_admit_obs          0.6286      0.257      2.447      0.014       0.125       1.132
initial_admit_elect        2.0345      0.243      8.389      0.000       1.559       2.510
const                    -68.0393      3.470    -19.605      0.000     -74.841     -61.237
==========================================================================================

Possibly complete quasi-separation: A fraction 0.79 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

```
#  Perform backward elimination #2 remove "Income" p=0.670
y = LogReg_df.ReAdmis
X = LogReg_df[["Children","Full_meals_eaten", "HighBlood", "Stroke", "Overweight", "Arthritis",
"Diabetes", "Hyperlipidemia", "BackPain", "Anxiety", "Asthma", "Initial_days", "initial_admit_obs",
"initial_admit_elect" ]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.039469
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9985
Method:                           MLE   Df Model:                           14
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                  0.9400
Time:                        15:35:34   Log-Likelihood:                -394.69
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
=======================================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------
Children                0.0783      0.041      1.888      0.059      -0.003       0.160
Full_meals_eaten        0.0818      0.093      0.877      0.380      -0.101       0.265
HighBlood               0.7683      0.194      3.960      0.000       0.388       1.149
Stroke                  1.4523      0.241      6.035      0.000       0.981       1.924
Overweight             -0.1426      0.205     -0.697      0.486      -0.544       0.259
Arthritis              -1.0884      0.201     -5.427      0.000      -1.482      -0.695
Diabetes                0.4095      0.206      1.985      0.047       0.005       0.814
Hyperlipidemia          0.3481      0.196      1.776      0.076      -0.036       0.732
BackPain                0.2525      0.186      1.355      0.175      -0.113       0.618
Anxiety                -0.8215      0.200     -4.098      0.000      -1.214      -0.429
Asthma                 -1.1439      0.210     -5.449      0.000      -1.555      -0.733
Initial_days            1.2403      0.063     19.638      0.000       1.116       1.364
initial_admit_obs       0.6233      0.256      2.431      0.015       0.121       1.126
initial_admit_elect     2.0276      0.242      8.383      0.000       1.554       2.502
const                 -67.9634      3.463    -19.624      0.000     -74.751     -61.175
=======================================================================================

Possibly complete quasi-separation: A fraction 0.79 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

```
#  Perform backward elimination # 3 remove "Full_meals_eaten" p=0.380
y = LogReg_df.ReAdmis
X = LogReg_df[["Children", "HighBlood", "Stroke", "Overweight", "Arthritis", "Diabetes",
"Hyperlipidemia", "BackPain", "Anxiety", "Asthma", "Initial_days", "initial_admit_obs",
"initial_admit_elect" ]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.039507
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9986
Method:                           MLE   Df Model:                           13
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                  0.9399
Time:                        15:36:47   Log-Likelihood:                -395.07
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Children                0.0790      0.041      1.908      0.056      -0.002       0.160
HighBlood               0.7650      0.194      3.946      0.000       0.385       1.145
Stroke                  1.4553      0.241      6.042      0.000       0.983       1.927
Overweight             -0.1499      0.205     -0.733      0.464      -0.551       0.251
Arthritis              -1.0828      0.200     -5.405      0.000      -1.475      -0.690
Diabetes                0.4138      0.206      2.008      0.045       0.010       0.818
Hyperlipidemia          0.3532      0.196      1.803      0.071      -0.031       0.737
BackPain                0.2498      0.186      1.342      0.180      -0.115       0.615
Anxiety                -0.8206      0.200     -4.098      0.000      -1.213      -0.428
Asthma                 -1.1349      0.209     -5.421      0.000      -1.545      -0.725
Initial_days            1.2399      0.063     19.638      0.000       1.116       1.364
initial_admit_obs       0.6289      0.256      2.456      0.014       0.127       1.131
initial_admit_elect     2.0364      0.242      8.425      0.000       1.563       2.510
const                 -67.8709      3.459    -19.621      0.000     -74.651     -61.091
==============================================================================

Possibly complete quasi-separation: A fraction 0.79 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

```
#  Perform backward elimination #4 remove "Overweight" p=0.464
y = LogReg_df.ReAdmis
X = LogReg_df[["Children", "HighBlood", "Stroke",  "Arthritis", "Diabetes", "Hyperlipidemia",
"BackPain", "Anxiety", "Asthma", "Initial_days", "initial_admit_obs", "initial_admit_elect"
]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.039534
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9987
Method:                           MLE   Df Model:                           12
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                  0.9399
Time:                        15:38:31   Log-Likelihood:                -395.34
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
Children                0.0801      0.041      1.935      0.053      -0.001       0.161
HighBlood               0.7539      0.193      3.904      0.000       0.375       1.132
Stroke                  1.4547      0.241      6.040      0.000       0.983       1.927
Arthritis              -1.0908      0.200     -5.454      0.000      -1.483      -0.699
Diabetes                0.4174      0.206      2.028      0.043       0.014       0.821
Hyperlipidemia          0.3508      0.196      1.793      0.073      -0.033       0.734
BackPain                0.2522      0.186      1.356      0.175      -0.112       0.617
Anxiety                -0.8200      0.200     -4.095      0.000      -1.212      -0.428
Asthma                 -1.1354      0.209     -5.426      0.000      -1.546      -0.725
Initial_days            1.2377      0.063     19.665      0.000       1.114       1.361
initial_admit_obs       0.6241      0.256      2.437      0.015       0.122       1.126
initial_admit_elect     2.0307      0.242      8.407      0.000       1.557       2.504
const                 -67.8490      3.455    -19.638      0.000     -74.621     -61.077
==============================================================================================

Possibly complete quasi-separation: A fraction 0.78 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

```
#  Perform backward elimination #5 remove "BackPain" p=0.175
y = LogReg_df.ReAdmis
X = LogReg_df[["Children", "HighBlood", "Stroke",  "Arthritis", "Diabetes", "Hyperlipidemia",
"Anxiety", "Asthma", "Initial_days", "initial_admit_obs", "initial_admit_elect" ]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.039627
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9988
Method:                           MLE   Df Model:                           11
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                  0.9397
Time:                        15:39:34   Log-Likelihood:                -396.27
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
=========================================================================================
                            coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------
Children                  0.0798      0.041      1.930      0.054      -0.001       0.161
HighBlood                 0.7602      0.193      3.941      0.000       0.382       1.138
Stroke                    1.4436      0.240      6.004      0.000       0.972       1.915
Arthritis                -1.0908      0.200     -5.463      0.000      -1.482      -0.699
Diabetes                  0.4154      0.206      2.016      0.044       0.012       0.819
Hyperlipidemia            0.3541      0.195      1.811      0.070      -0.029       0.737
Anxiety                  -0.7947      0.199     -3.997      0.000      -1.184      -0.405
Asthma                   -1.1342      0.209     -5.427      0.000      -1.544      -0.725
Initial_days              1.2341      0.063     19.705      0.000       1.111       1.357
initial_admit_obs         0.6382      0.256      2.498      0.012       0.137       1.139
initial_admit_elect       2.0377      0.241      8.446      0.000       1.565       2.511
const                   -67.5631      3.433    -19.682      0.000     -74.291     -60.835
=========================================================================================

Possibly complete quasi-separation: A fraction 0.78 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

```
#  Perform backward elimination #6 remove "Hyperlipidemia" p=0.070
y = LogReg_df.ReAdmis
X = LogReg_df[["Children", "HighBlood", "Stroke",  "Arthritis", "Diabetes", "Anxiety", "Asthma",
"Initial_days", "initial_admit_obs", "initial_admit_elect" ]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.039792
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9989
Method:                           MLE   Df Model:                           10
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                 0.9395
Time:                        15:40:38   Log-Likelihood:                -397.92
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
=========================================================================================
                            coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------
Children                  0.0758      0.041      1.839      0.066      -0.005       0.157
HighBlood                 0.7634      0.192      3.966      0.000       0.386       1.141
Stroke                    1.4191      0.239      5.930      0.000       0.950       1.888
Arthritis                -1.0852      0.199     -5.445      0.000      -1.476      -0.695
Diabetes                  0.3947      0.205      1.925      0.054      -0.007       0.797
Anxiety                  -0.8009      0.198     -4.036      0.000      -1.190      -0.412
Asthma                   -1.1311      0.209     -5.415      0.000      -1.540      -0.722
Initial_days              1.2292      0.062     19.747      0.000       1.107       1.351
initial_admit_obs         0.6473      0.255      2.541      0.011       0.148       1.147
initial_admit_elect       2.0453      0.241      8.503      0.000       1.574       2.517
const                   -67.1728      3.405    -19.729      0.000     -73.846     -60.499
=========================================================================================

Possibly complete quasi-separation: A fraction 0.78 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

```
#  Perform backward elimination #7 remove "Children" p=0.066
y = LogReg_df.ReAdmis
X = LogReg_df[["HighBlood", "Stroke",  "Arthritis", "Diabetes", "Anxiety", "Asthma", "Initial_days",
"initial_admit_obs", "initial_admit_elect" ]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.039962
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9990
Method:                           MLE   Df Model:                            9
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                  0.9392
Time:                        15:41:26   Log-Likelihood:                -399.62
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==========================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
HighBlood                  0.7865      0.192      4.098      0.000       0.410       1.163
Stroke                     1.4082      0.239      5.903      0.000       0.941       1.876
Arthritis                 -1.0842      0.199     -5.450      0.000      -1.474      -0.694
Diabetes                   0.3907      0.204      1.915      0.055      -0.009       0.791
Anxiety                   -0.7865      0.198     -3.980      0.000      -1.174      -0.399
Asthma                    -1.1471      0.208     -5.515      0.000      -1.555      -0.739
Initial_days               1.2278      0.062     19.738      0.000       1.106       1.350
initial_admit_obs          0.6535      0.254      2.569      0.010       0.155       1.152
initial_admit_elect        2.0437      0.240      8.510      0.000       1.573       2.514
const                    -66.9455      3.398    -19.704      0.000     -73.605     -60.286
==========================================================================================

Possibly complete quasi-separation: A fraction 0.78 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

```
#  Perform backward elimination #8 remove "Diabetes" p=0.055
y = LogReg_df.ReAdmis
X = LogReg_df[["HighBlood", "Stroke",  "Arthritis", "Anxiety", "Asthma", "Initial_days",
"initial_admit_obs", "initial_admit_elect" ]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.040147
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9991
Method:                           MLE   Df Model:                            8
Date:                Sat, 08 Jun 2024   Pseudo R-squ.:                 0.9389
Time:                        15:42:14   Log-Likelihood:                -401.47
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
HighBlood              0.7359      0.189      3.887      0.000       0.365       1.107
Stroke                 1.3597      0.236      5.752      0.000       0.896       1.823
Arthritis             -1.0660      0.198     -5.385      0.000      -1.454      -0.678
Anxiety               -0.7963      0.197     -4.033      0.000      -1.183      -0.409
Asthma                -1.1402      0.208     -5.492      0.000      -1.547      -0.733
Initial_days           1.2228      0.062     19.780      0.000       1.102       1.344
initial_admit_obs      0.6291      0.253      2.485      0.013       0.133       1.125
initial_admit_elect    2.0317      0.239      8.501      0.000       1.563       2.500
const                -66.5296      3.368    -19.754      0.000     -73.131     -59.929
==============================================================================

Possibly complete quasi-separation: A fraction 0.78 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

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

1. The initial logistic regression model contained 19 variables that included information about patient demographics, patients cost for hospital stay, and illnesses/issues that may or may not have been part of the reason a patient was in the hospital. After checking for variance inflation initially and then using a stepwise (backwards) elimination based on p values, the reduced logistic regression model contains 8 variables that include patient conditions, number of days during initial admit and reasons around initial admit.
   We can further analyze the data set using train test split method to determine the portion of correctly predicted instances (accuracy), the portion of true positive instances out of all predicted instances (precision), the portion of true positive instances out of all actual instances (recall), the harmonic mean of precision and recall (F1 score), the area under the receiver operating curve which shows if the model is able to distinguish between classes (ROC AUC) and a matrix showing true positives, true negative, false positives and false negatives (confusion matrix) which are calculated using Scikit-Learn Developers. (n.d.).

   Additionally the confusion matrix identifies where a model may get 'confused' or not indicate the correct outcome. We are using a binary class dataset for the independent variable, ReAdmis. Kundu (2022) states that a confusion matrix is a succinct and organized way of getting deeper information about a classifier which is computed by mapping the expected outcomes to the predicted outcomes of a model.

2. The output and *all* calculations of the above analysis performed are listed below.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix

# Assuming 'LogReg_df' is your DataFrame and 'ReAdmis' is the target variable
X = LogReg_df[["HighBlood", "Stroke", "Arthritis", "Anxiety", "Asthma", "Initial_days", "initial_admit_obs", "initial_admit_elect"]]
y = LogReg_df["ReAdmis"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the logistic regression model
model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)

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

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print the evaluation metrics
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"ROC AUC: {roc_auc}")
print(f"Confusion Matrix:\n{conf_matrix}")
```

Results:

Accuracy: 0.986
This indicates that 98.6% of the predictions made by the model are correct.

Precision: 0.984352773826458
When the model predicts a readmission, it is accurate 98.4% of the time.

Recall: 0.9760225669957687
Recall of 97.6% indicates that the model identifies that percentage of actual readmissions.

F1 Score: 0.980169971671388
An F1 score of 98% indicates that the model balances precision and recall and accurately predicts at a high level.

ROC AUC: 0.9992428869060951
With 99.9% falling under the receiver curve, the model is excellent at distinguishing between patients that will be readmitted and those that won't.

Confusion Matrix:
[[1280 11]
[17 692]]

True Positives: 692	Patients correctly identified as readmitted.
True Negatives: 1280	Patients correctly identified as not readmitted
False Positives: 11	Patients incorrectly identified as readmitted.
False Negatives: 17	Patients incorrectly identified as not readmitted.

3. An executable error-free copy of the code is included. File name D208_Austin_T_Task2.ipynb

Part V: Data Summary and Implications

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

1. Discuss the results of your data analysis, including the following elements:

- In order to create a regression equation for the reduced model, information from Dash (2022) was used. The article describes how to derive the logistic regression equation as shown below:

The general form of the logistic regression equation is:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \ldots + \beta_n \cdot X_n$$

Where p is the probability of readmission occurring, B_0 is the intercept, B_1 are the coefficients for the predictor variables HighBlood, Stroke, Arthritis, Anxiety, Asthma, Initial_days, initial_admit_obs, and initial_admit_elect.

The final equation is:
Log(p/1-p) = -66.5296 + 0.7359(HighBlood) + 1.3519(Stroke) - 1.0660(Arthritis) - 0.7963(Anxiety) – 1.1402(Asthma) + 1.2286(Initial_days) + 0.6291(initial_admit_obs) + 0.2317(initial_admit_elec)

- The remaining variables in the reduced model are HighBlood, Stroke, Arthritis, Anxiety, Asthma, Initial_days, initial_admit_obs, and initial_admit_elect. The variable they are being tested against is ReAdmis, which indicates yes or no to the question if a patient was readmitted within 30 days of their initial hospitalization. An interpretation of each of their coefficients is as follows:
    1. Intercept (-66.5296) is the log-odds of readmission when all predictor variables are zero. It is not very interpretable on its own but is essential for the model.
    2. HighBlood (0.7359) shows that patients with high blood pressure have an increase of being readmitted by 0.7359 compared to patients without this disease. This translates to an odds ratio of approximately 2.087 meaning patients with high blood pressure are 2.087 times more likely to be readmitted.
    3. Stroke (1.3519) identifies patients who have had a stroke have an odds ratio of approximately 3.863 so they are 3.863 times more likely to be readmitted versus those without a history of stroke.

4. Arthritis (-1.0660) indicates an odds ratio of 0.344 meaning patients with arthritis are about .0344 times as likely to be readmitted or equivalently they are less likely to be readmitted.
5. Anxiety (-0.7963) this log-odds value translates to an odds ratio of about 0.451 showing that patients with anxiety are about 0.451 times as likely to be readmitted.
6. Asthma (-1.1402) suggests based on odds ratio of 0.320 that patients with asthma are that much more likely to be readmitted.
7. Initial_days (1.2286) has an increased value of 1.2286 for each additional initial day a patient spends in the hospital and an odds ratio of 3.416.
8. initial_admit_obs (0.6291) also shows an increased value of 0.6291 for patients admitted for observation and a odds ratio of 1.876 times more likely to be readmitted if they were initially admitted for observation.
9. initial_admit_elect (0.2317) is lower than if a patient were admitted for observation. Patients who are admitted electively are 1.261 times more likely to be readmitted.

HighBlood, Stroke, Initial_days, initial_admit_obs and initial_admit_elec are considered positive coefficients and increase the likelihood of admission.
Whereas, Arthritis, Anxiety, and Asthma have negative coefficients and decrease the likelihood of admission.

- The reduced logistical regression model shows both statistical and practical significance. The predicitor variables in the model show statistically significant based on the fact that their p values are less than 0.05.
  We can also use McFadden's R-Squared test which is a pseudo R-squared measure that indicates a measure of the model's goodness-of-fit where values closer to one indicate a better fit (Shafrin, 2016).

  Based on the code below, we can calculate the value and determine how fit the model is.

```
# Calculate McFadden's R-squared
X = LogReg_df[["HighBlood", "Stroke", "Arthritis", "Anxiety", "Asthma", "Initial_days",
"initial_admit_obs", "initial_admit_elect"]]
y = LogReg_df["ReAdmis"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fit the logistic regression model
logit_model = sm.Logit(y_train, sm.add_constant(X_train)).fit(disp=0)

# Predict on the test set
y_prob = logit_model.predict(sm.add_constant(X_test))
y_pred = y_prob >= 0.5

ll_null = logit_model.llnull
ll_model = logit_model.llf
mcfadden_r2 = 1 - (ll_model / ll_null)

# Print the results
print(f"McFadden's R-squared: {mcfadden_r2}")
```

McFadden's R-squared: 0.9367996620563905

This value indicates that the model explains a large portion of the variance. By combining the goodness-of-fit metric with the previously observed evaluation metrics of accuracy, precision, recall, F1, and ROC AUC we can conclude that the reduced model is both statistically and practically significant.

- There are still some limitations of the data analysis such as data quality issues that may be caused from inaccurately recorded information. Since much of this data is recorded when a patient is being admitted and not based on tests run while in the hospital, there could be errors in gathering and/or inputting the data. There could also be some information that is not accurately known by those giving the information at the time of admission. We are also only looking at one window of readmission time, within 30 days. Some patients are more likely to be compliant within the first month or two and may be readmitted on day 31 or day 60 based on why they were initially in the hospital.
  There are still too many factors that are unknown to fully trust the variables as being truly meaningful to this research. Based on what we know, we have provided the best analysis.

2. The recommended course of action, based on the above results would be for healthcare professionals to leverage the model to enhance patient care and reduce readmission rates. We have been able to show that there are factors that are statistically significant and practically significant such as high blood pressure, initial days in the hospital and stroke that can be used to predict if a patient will be readmitted. The model can be used to identify patients at high risk of readmission, allowing for targeted interventions. Focus on patients with these conditions or with longer initial hospitalizations will help reduce readmission. The significant predictors (e.g., high blood pressure, stroke, initial days) can inform clinical and administrative decisions to improve patient outcomes. Ensuring that patients go home with enough medications, making sure that patients keep follow up appointments or even having staff that reach out to check on patients could make a difference in readmissions. Finally, hospitals can allocate resources more effectively by focusing on high-risk patients identified by the model. If a patient has multiple of these predictors aligned with their care, that may mean that additional resources around training and education of not only the patient but their families about these factors could help reduce readmissions.

Part VI: Demonstration

G. Panopto video recording

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=b160ba9b-adc5-4995-a30a-b18c00dae32f

H. Web Sources

Scikit-Learn Developers. (n.d.). *Logistic Regression*. Retrieved from https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

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

Sources

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

Dash, S. (2022). *Understanding logistic regression: The odds ratio, sigmoid, MLE, et al*. Towards Data Science. Retrieved June 8, 2024, from https://towardsdatascience.com/understanding-logistic-regression-the-odds-ratio-sigmoid-mle-et-al-740cebf349a3

Shafrin, J. (2016, December 28). *What is a pseudo R-squared?* Retrieved June 8, 2024, from https://www.healthcare-economist.com/2016/12/28/what-is-a-pseudo-r-squared/

Kundu, R. (2022, September 13). *Confusion matrix guide: Understand and implement confusion matrices in machine learning*. V7 Labs. Retrieved June 8, 2024, from https://www.v7labs.com/blog/confusion-matrix-guide

Neter, J., Kutner, M. H., Nachtsheim, C. J., & Wasserman, W. (1996). *Applied Linear Statistical Models* (4th ed.). McGraw-Hill.

Sharma, K. (2020, December 5). Mastering statistical analysis with StatsModel library with code. Level Up Coding. Retrieved June 7, 2024, from https://levelup.gitconnected.com/mastering-statistical-analysis-with-statsmodel-library-with-code-d59e84c5b371

Statistics Solutions. (2010, December 20). *Assumptions of logistic regression*. Retrieved June 6, 2024, from https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-logistic-regression/

Swaminathan, S. (2018). Logistic regression detailed overview. *Towards Data Science*. Retrieved from [https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc](https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

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

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