Part I

A. Question

A.1. Research Question

Is a patient's length of initial hospitalization correlated to any preexisting conditions in this data set as demonstrated by multiple linear regression modeling?

A.2. Goals

The goals of this analysis is to demonstrate the correlation between any of the patient's preexisting conditions in the data set and the patient's length of hospitalization. Variables identified as preexhisting conditions include: the patient's age, their socioeconomic status as reflected by their income, the patient's gender, if the patient was readmitted, if the patient has high blood pressure, has had a stroke, is overweight, has arthritis, diabetes, hyperlipidemia, back pain, anxiety, allergic rhinitus, acid reflux, or asthma. The correlation will then be assessed to determine if any business insight can be learned from the correlations. Through the identification of variables that put a patient at increase risk for prolonged hospitalization we may be able to identify patients and provide intervention to decrease their hospital stay.

Part II

B. Justification

B.1. Assumptions

To create a multiple linear regression model, we must make four assumptions. The first assumption is that the relationships between the dependent (y) variable and all of the independent (x) variables are linear. The next assumption is that observations are independent from one another and random. This means the observations must not be choosen to sway the results. An example of this would be choosing to not take observations from a very ill patient who recovered quickly because we were worried that this may not reflect well in our study results. We would still need to include this observation. A third assumption is that the independent variables do not highly correlate to one another. An example of this would be if all patients with high blood pressure were all older patients. These two variables are no longer independent from each other so we would not want to include both variables in our regression analysis. The fourth assumption is that the explanatory power increases with an increasing number of independent variables. If the explanatory power does not increase when a variable is add then it should not be included in the model as either it is correlated to an existing variable or it has little impact on the model.

B.2. Benefits

The two main benefits of choosing python over R are my familiarity with python and broader application power of python. While the presentation capabilities of R are nice, I chose to do this project in python because I feel as though I am more comfortable with python. I also appreciate that python can be used for much more than just statistics.

B.3. Explanation

Multiple linear regression modeling is appropriate for this dataset because all of the above assumption are met. In addition, since we are attempting to determine if multiple variables are correlated with admission length, multiple linear regression is a good tool to do so.

Part III
C. Data Prep

C.1. Data cleaning steps-

My data cleaning goal was to get data to apply to a multiple linear regression model. I started by importing packages and loading my data set.

```
# import packages
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import linear_model

#import file from mapped working directory
os.chdir('C:/Users/Whit/Documents/Data')
medical_df = pd.read_csv('medical_raw_data.csv',dtype={'locationid':np.int64})

# view file data
medical_df.info()
```

I simplified the data set down to only the variables of interest and assessed the data set for duplicates, of which none were returned. Duplicates can impact the correlation of variable and may make independent variables appear more or less correlated with the dependent variable than they actually are.

```
#Simplify dataset to only variables of interest
medical_df = medical_df[['Initial_days','Age', 'Overweight', 'HighBlood', 'Stroke', 'Arthritis',
'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis',
'Asthma']]

#detect duplicates
medical_df.duplicated()
print(medical_df.duplicated().value_counts())
```

Then I ran isna().sum() and returned the count of missing values for each column. With missing values, I would not have been able to run the multiple linear regression model. The dataset demonstrated no missing values.

```
#View missing data
medical_df.isna().sum()

#view histograms of data
medical_df.hist()
```

Next, Age, and Initial_days were reviewed for outliers. Outliers can impact the correlation between variables, so it is appropriate to remove them now so they do not impact the model. No outliers were found.

```
#review for outliers
medical_df.boxplot(['Age'])

#review for outliers
medical_df.boxplot(['Initial_days'])
```

C.2. Variables

The dependent variable is Initial_days, this is a quantitative, continuous or ratio data. The independent or explanatory variables are Age, HighBlood, Stroke, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagistis, and Asthma. Age is quantitative, continuous data. HighBlood, Stroke, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagistis, and Asthma are categorical, nominal data.

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

```
count    10000.000000
mean        34.433651
std         24.860232
min          1.001981
25%          8.928987
50%         34.446941
75%         59.459981
max         71.981486
Name: Initial_days, dtype: float64
```

```
medical_df['HighBlood_Yes'].describe()
```

```
count    10000.000000
mean         0.409000
std          0.491674
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          1.000000
Name: HighBlood_Yes, dtype: float64
```

```
Name: Age, dtype: float64
```

```
Name: Stroke_Yes, dtype: float64
```

```
medical_df['Overweight'].describe()
```

```
medical_df['Arthritis_Yes'].describe()
```

```
medical_df['Diabetes_Yes'].describe()
```

```
count    10000.00000
mean         0.27380
std          0.44593
min          0.00000
25%          0.00000
50%          0.00000
75%          1.00000
max          1.00000
Name: Diabetes_Yes, dtype: float64
```

```
medical_df['Allergic_rhinitis_Yes'].describe()
```

```
count    10000.000000
mean         0.394100
std          0.488681
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          1.000000
Name: Allergic_rhinitis_Yes, dtype: float64
```

```
medical_df['Hyperlipidemia_Yes'].describe()
```

```
count    10000.000000
mean         0.337200
std          0.472777
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          1.000000
Name: Hyperlipidemia_Yes, dtype: float64
```

```
medical_df['Reflux_esophagitis_Yes'].describe()
```

```
count    10000.000000
mean         0.413500
std          0.492486
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          1.000000
Name: Reflux_esophagitis_Yes, dtype: float64
```

```
medical_df['BackPain_Yes'].describe()
```

```
count    10000.000000
mean         0.411400
std          0.492112
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          1.000000
Name: BackPain_Yes, dtype: float64
```

```
medical_df['Asthma_Yes'].describe()
```

```
count    10000.00000
mean         0.28930
std          0.45346
min          0.00000
25%          0.00000
50%          0.00000
75%          1.00000
max          1.00000
Name: Asthma_Yes, dtype: float64
```

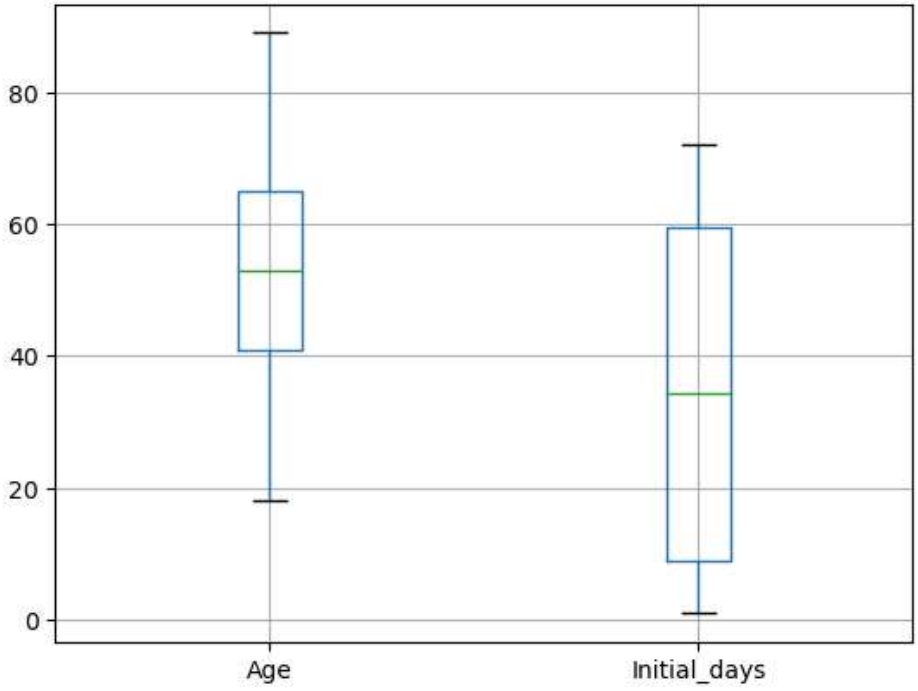
```
medical_df['Anxiety'].describe()
```

```
count    10000.000000
mean         0.290600
std          0.454062
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          1.000000
Name: Anxiety, dtype: float64
```

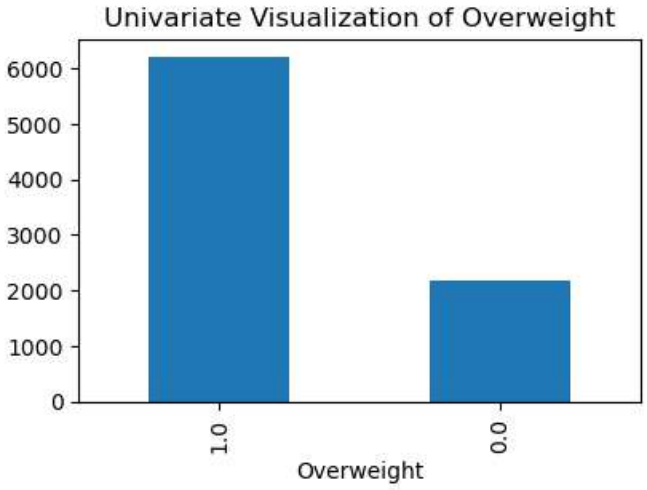
C.3.Univariate and Bivariate visualization

#review for outliers
medical_df[['Age', 'Initial_days']].boxplot

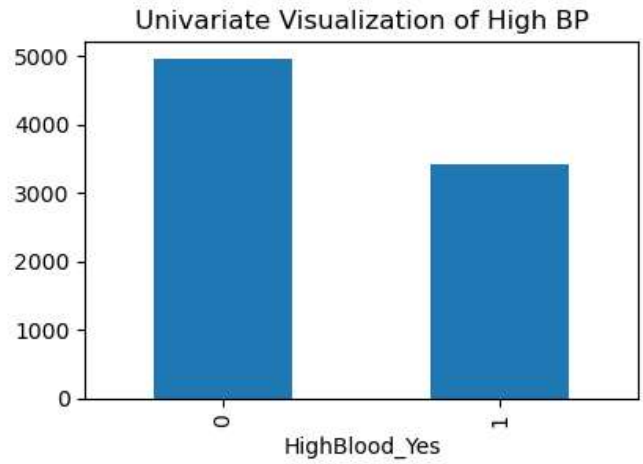
Univariate Analysis of Age and Initial_days



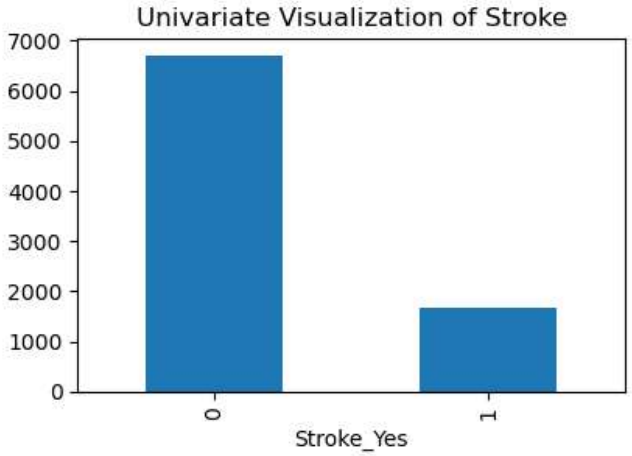

```
# Plot accidents depending on type
fig = plt.figure(figsize=(10,10))
fig_dims = (3, 2)

plt.subplot2grid(fig_dims, (0, 0))
medical_df['Overweight'].value_counts().plot(kind='bar',
                                             title='Univariate Visualization of Overweight')
```



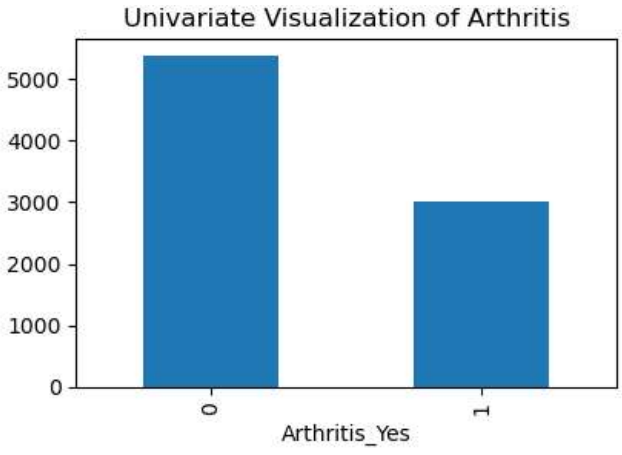

```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['HighBlood_Yes'].value_counts().plot(kind='bar',
                                             title='Univariate Visualization of High BP')
```



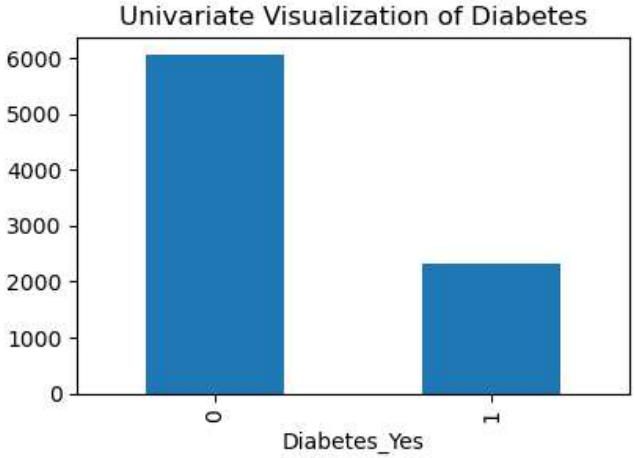

```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['Stroke_Yes'].value_counts().plot(kind='bar',
                                  title='Univariate Visualization of Stroke')
```



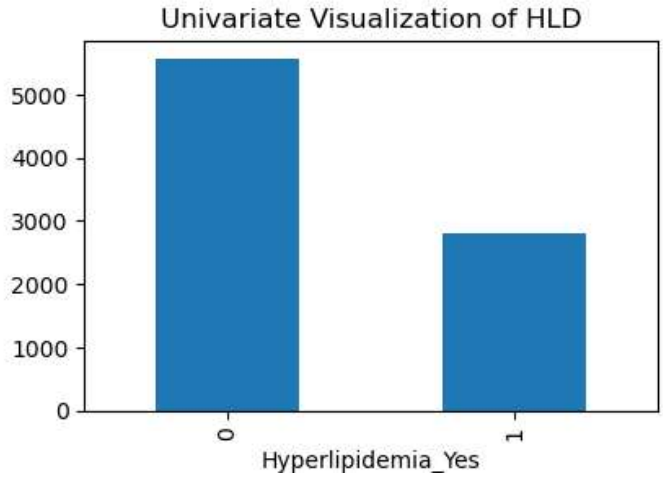

```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['Arthritis_Yes'].value_counts().plot(kind='bar',
                                  title='Univariate Visualization of Arthritis')
```



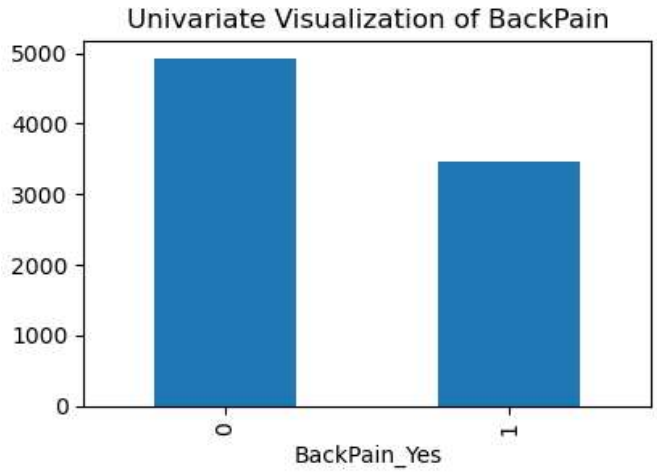
```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['Diabetes_Yes'].value_counts().plot(kind='bar',
                                title='Univariate Visualization of Diabetes')
```



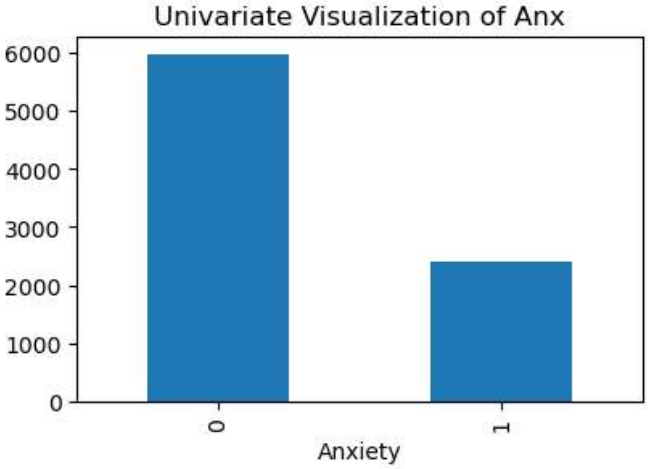

```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['Hyperlipidemia_Yes'].value_counts().plot(kind='bar',
                                title='Univariate Visualization of HLD')
```



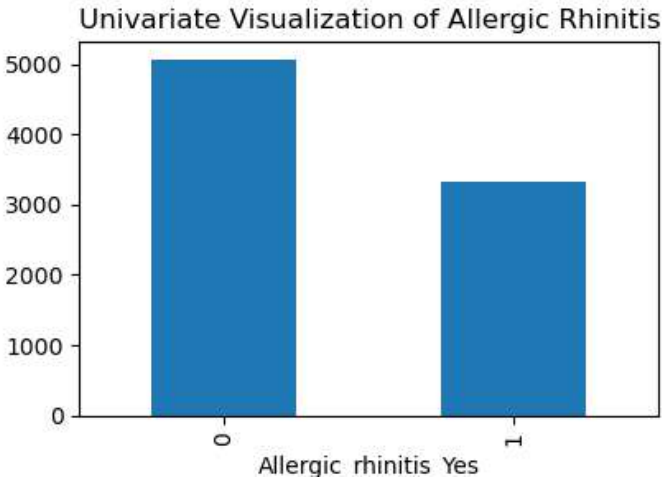

```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['BackPain_Yes'].value_counts().plot(kind='bar',
                                title='Univariate Visualization of BackPain')
```



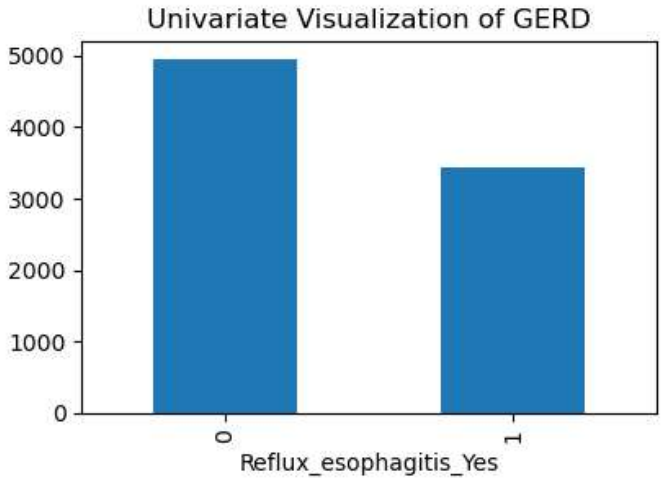

```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['Anxiety'].value_counts().plot(kind='bar',
                                          title='Univariate Visualization of Anx')
```



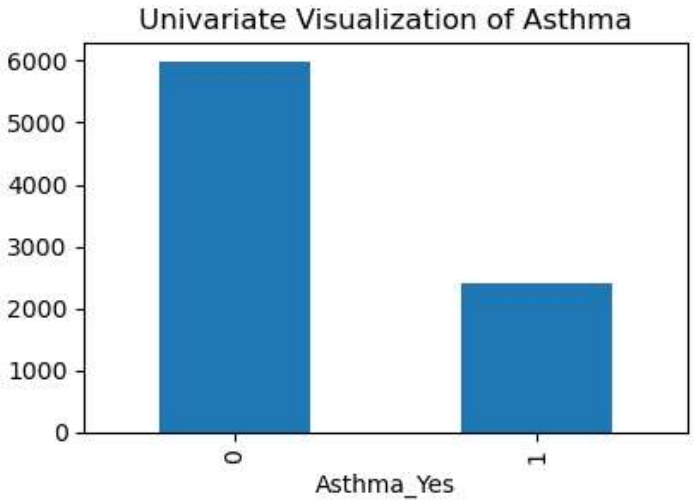

```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['Allergic_rhinitis_Yes'].value_counts().plot(kind='bar',
                                          title='Univariate Visualization of Allergic Rhinitis')
```




```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['Reflux_esophagitis_Yes'].value_counts().plot(kind='bar',
                                  title='Univariate Visualization of GERD')
```

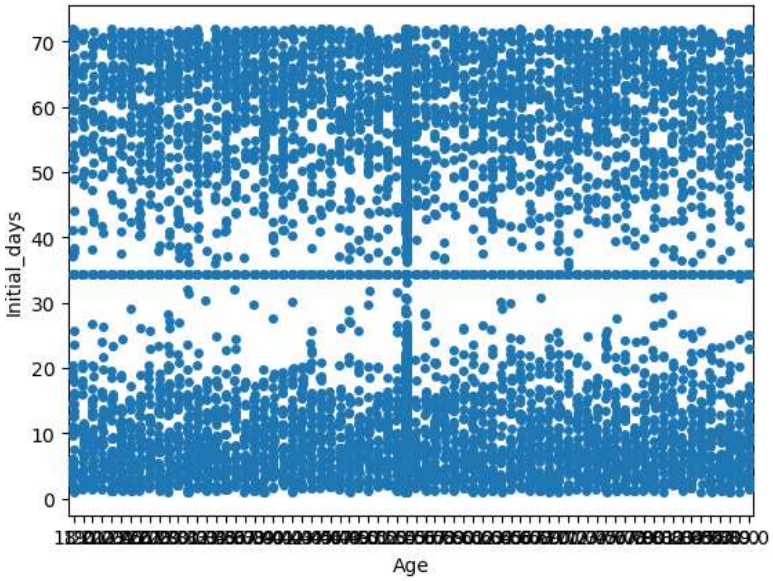



```
fig = plt.figure(figsize=(10,10))
plt.subplot2grid(fig_dims, (0, 0))
medical_df['Asthma_Yes'].value_counts().plot(kind='bar',
                                  title='Univariate Visualization of Asthma')
```

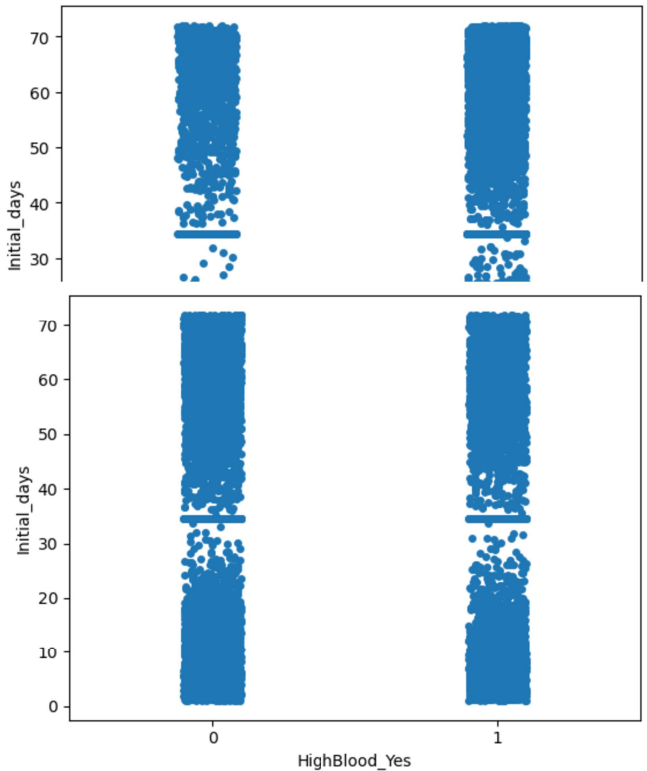


Bivariate Visualization of income vs dependent variables

sns.stripplot(x="Age", y="Initial_days", data=medical_df)

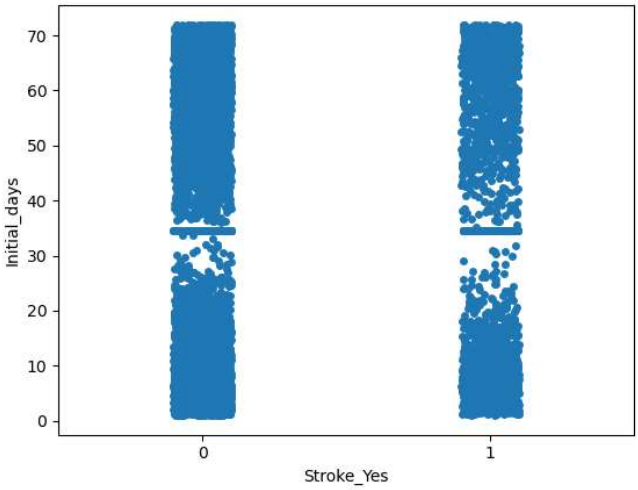


sns.stripplot(x="Overweight", y="Initial_days", data=medical_df)

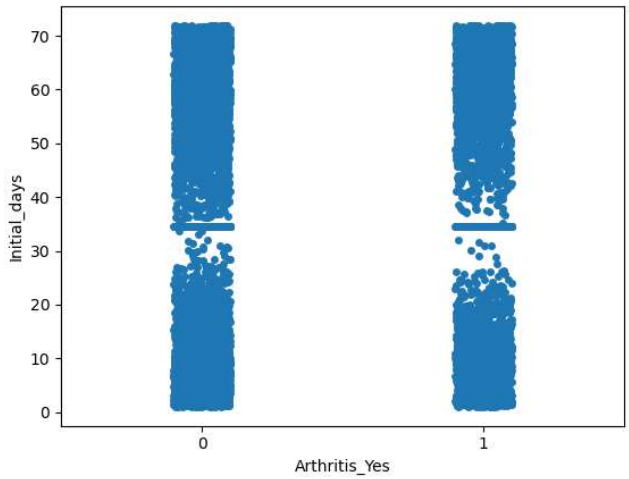


sns.stripplot(x="HighBlood_Yes", y="Initial_days", data=medical_df)

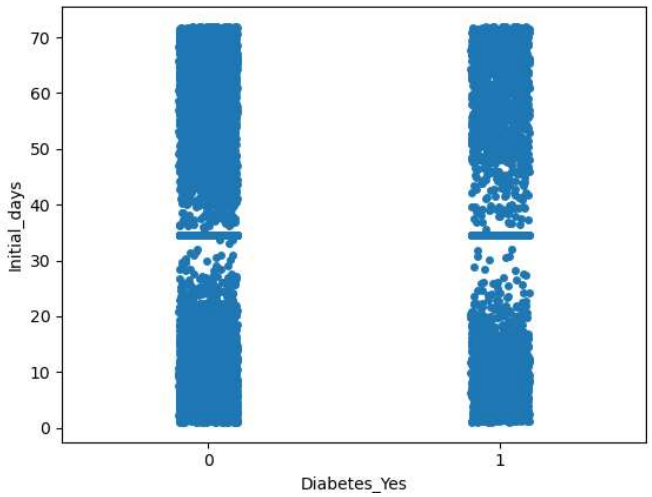
sns.stripplot(x="Stroke_Yes", y="Initial_days", data=medical_df)



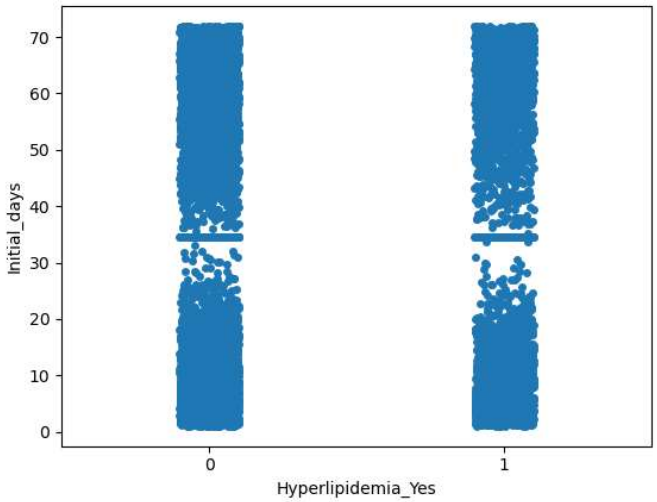
sns.stripplot(x="Arthritis_Yes", y="Initial_days", data=medical_df)



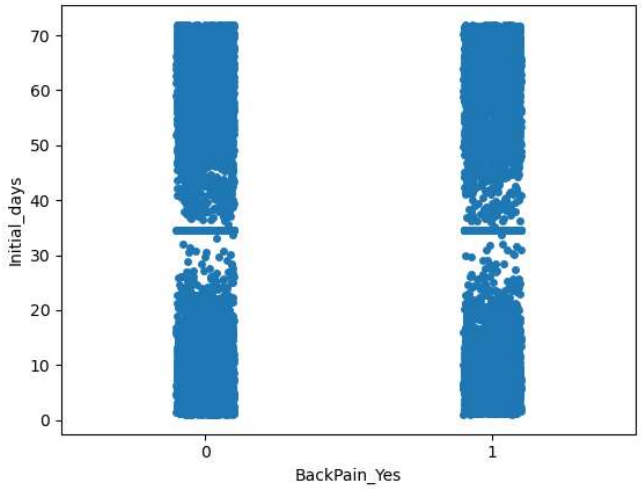
sns.stripplot(x="Diabetes_Yes", y="Initial_days", data=medical_df)



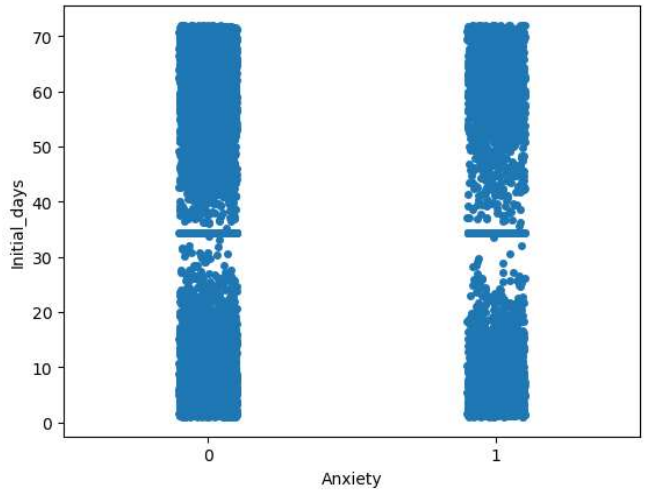
sns.stripplot(x="Hyperlipidemia_Yes", y="Initial_days", data=medical_df)



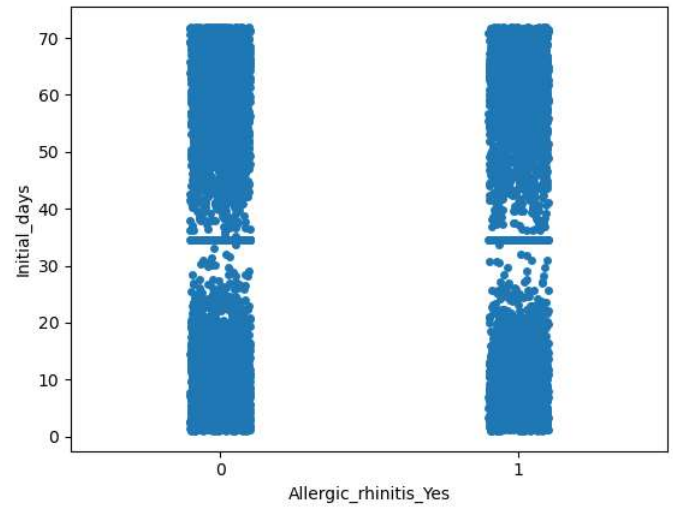
sns.stripplot(x="BackPain_Yes", y="Initial_days", data=medical_df)



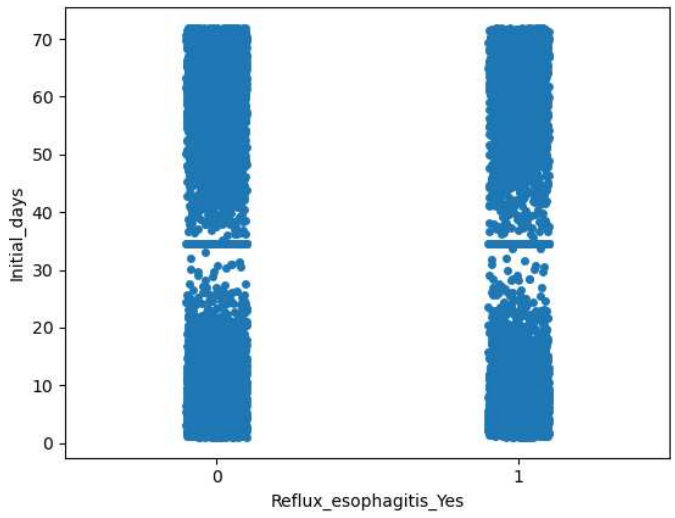
sns.stripplot(x="Anxiety", y="Initial_days", data=medical_df)



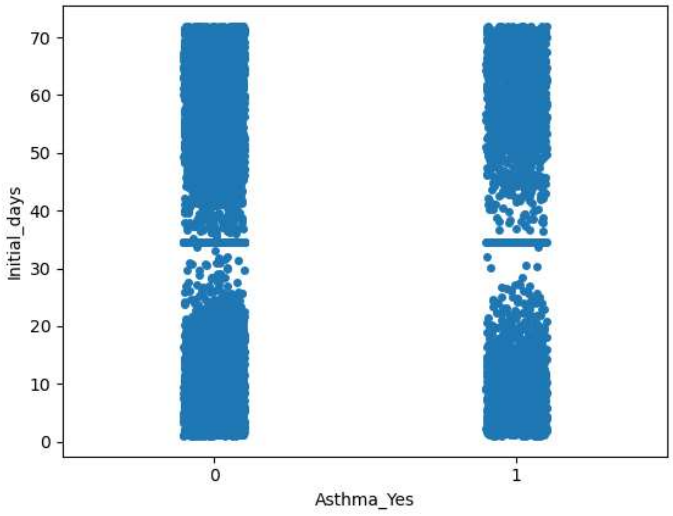
sns.stripplot(x="Allergic_rhinitis_Yes", y="Initial_days", data=medical_df)



sns.stripplot(x="Reflux_esophagitis_Yes", y="Initial_days", data=medical_df)



```
sns.stripplot(x="Asthma_Yes", y="Initial_days", data=medical_df)
```



C.4. Data Transformation

To perform multiple linear regression, all variables of interest need to be numeric or integers so all categorical variables of interest were transformed into integers via one hot encoding.

```
#One Hot Encoding for Categorical Columns
dummy_df = pd.get_dummies(medical_df, columns = ['HighBlood', 'Stroke', 'Arthritis',
        'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis' ,
        'Reflux_esophagitis', 'Asthma'], dtype=int)
dummy_df.info()

#Drop extra columns and rename
medical_df = dummy_df.drop(['HighBlood_No', 'Stroke_No', 'Arthritis_No','Diabetes_No',
        'Hyperlipidemia_No', 'BackPain_No', 'Anxiety_0.0', 'Allergic_rhinitis_No',
        'Reflux_esophagitis_No', 'Asthma_No'], axis=1)
medical_df = medical_df.rename(columns={'ReAdmis_Yes': 'ReAdmit'})
medical_df = medical_df.rename(columns={'Anxiety_1.0': 'Anxiety'})
medical_df.info()
```

C.5. CSV file

```
#Cleaned dataset
medical_df.to_csv('clean_medical_D208.csv')
```

Please see uploaded CSV file.

Part IV

D. Comparison

D.1. Linear Regression Model of Independent Variables

```
#OLS
medical_df['intercept']=1
```

```
lm_days = sm.OLS(medical_df['Initial_days'], medical_df[['Age', 'Overweight',
        'HighBlood_Yes', 'Stroke_Yes', 'Arthritis_Yes', 'Diabetes_Yes', 'Hyperlipidemia_Yes',
        'BackPain_Yes', 'Anxiety', 'Allergic_rhinitis_Yes', 'Reflux_esophagitis_Yes',
        'Asthma_Yes', 'intercept']]).fit()
print(lm_days.summary())
```

OLS Regression Results

Dep. Variable:	Initial_days	R-squared:	0.002
Model:	OLS	Adj. R-squared:	0.001
Method:	Least Squares	F-statistic:	1.445
Date:	Fri, 05 Jul 2024	Prob (F-statistic):	0.137
Time:	16:18:16	Log-Likelihood:	-46313.
No. Observations:	10000	AIC:	9.265e+04
Df Residuals:	9987	BIC:	9.275e+04
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Age	0.0111	0.014	0.801	0.423	-0.016	0.038
Overweight	-0.7636	0.566	-1.350	0.177	-1.872	0.345
HighBlood_Yes	-0.7350	0.506	-1.453	0.146	-1.727	0.257
Stroke_Yes	0.0682	0.622	0.110	0.913	-1.152	1.288
Arthritis_Yes	0.9063	0.519	1.746	0.081	-0.111	1.924
Diabetes_Yes	-0.1089	0.558	-0.195	0.845	-1.202	0.984
Hyperlipidemia_Yes	-0.0178	0.526	-0.034	0.973	-1.049	1.013
BackPain_Yes	0.8497	0.505	1.681	0.093	-0.141	1.840
Anxiety	0.0793	0.548	0.145	0.885	-0.994	1.153
Allergic_rhinitis_Yes	0.5002	0.509	0.983	0.326	-0.497	1.498
Reflux_esophagitis_Yes	0.7347	0.505	1.455	0.146	-0.255	1.725
Asthma_Yes	-1.0338	0.548	-1.885	0.059	-2.109	0.041
intercept	33.8324	1.042	32.478	0.000	31.790	35.874

Omnibus:	49574.832	Durbin-Watson:	0.364
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1077.279
Skew:	0.075	Prob(JB):	1.18e-234
Kurtosis:	1.399	Cond. No.	251.

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

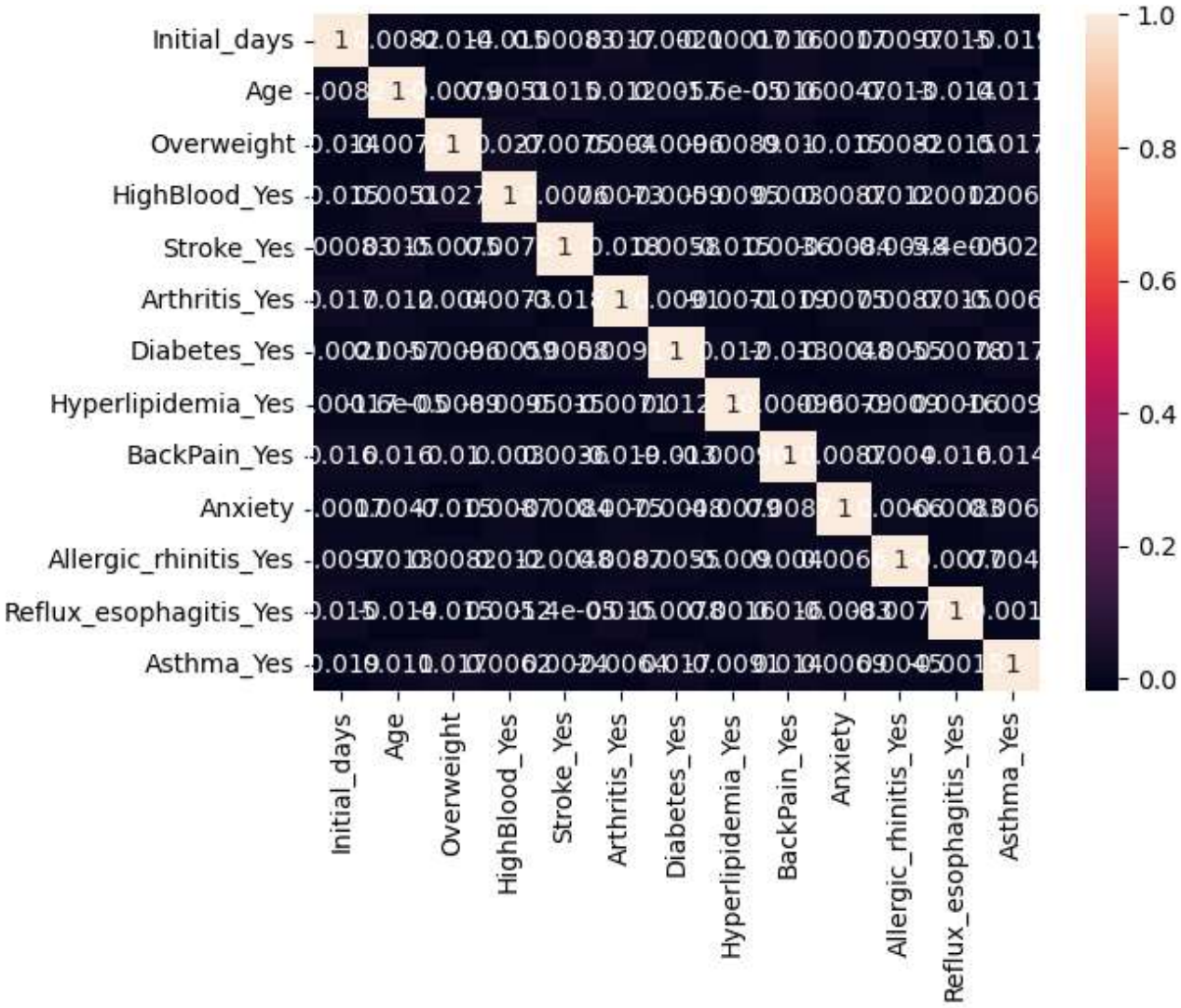
D.2.

Justification of Reduction

From the above linear regression model, we can see that all P-values are greater than 0.05, so all independent variables have a low impact on the dependent variable. In order to provide a reduced model I will use a P-value of 0.1, these variables have some impact on the dependent variable. In this case Age, Overweight, Stroke, Diabetes, Hyperlipidemia, Anxiety, and Allergic_rhinitis, all have a p-value greater than 0.1 so they will be removed.

Next, I will look at a heatmap to see if any variables share a high correlation.

```
#Heatmap
sns.heatmap(medical_df.corr(), annot=True)
plt.show()
```



As demonstrated by the heatmap, there is not a high correlation between any of the chosen variables. Based on this no variables need to be excluded.

Finally I looked at the variable independence factor.

```
#Determine VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
X=medical_df[['Age', 'Overweight', 'HighBlood_Yes', 'Stroke_Yes', 'Arthritis_Yes',
        'Diabetes_Yes', 'Hyperlipidemia_Yes', 'BackPain_Yes', 'Anxiety', 'Allergic_rhinitis_Yes',
        'Reflux_esophagitis_Yes', 'Asthma_Yes']]
vif_score=pd.DataFrame()
vif_score["feature"]=X.columns
vif_score['VIF']=[variance_inflation_factor(X.values, i) for i in range(len(X.columns))]
vif_score['VIF']=round(vif_score['VIF'],2)
```

```
vif_score=vif_score.sort_values(by="VIF", ascending = False)
print(vif_score)
```

	feature	VIF
0	Age	5.00
1	Overweight	3.20
7	BackPain_Yes	1.65
2	HighBlood_Yes	1.64
10	Reflux_esophagitis_Yes	1.63
9	Allergic_rhinitis_Yes	1.60
4	Arthritis_Yes	1.52
6	Hyperlipidemia_Yes	1.46
8	Anxiety	1.38
11	Asthma_Yes	1.38
5	Diabetes_Yes	1.35
3	Stroke_Yes	1.23

This again verifies that no variables share a high co-linearity, so none of the variables need to be removed based on this.

D.3. Reduced Linear Regression Model

Due to the above analysis, the model stands as:

```
#Reduced OLS model
medical_df['intercept']=1
lm_days_red = sm.OLS(medical_df['Initial_days'], medical_df[['intercept', 'HighBlood_Yes',
        'Arthritis_Yes', 'BackPain_Yes', 'Reflux_esophagitis_Yes', 'Asthma_Yes']]).fit()
print(lm_days_red.summary())
```

OLS Regression Results

Dep. Variable:	Initial_days	R-squared:	0.001
Model:	OLS	Adj. R-squared:	0.001
Method:	Least Squares	F-statistic:	2.764
Date:	Fri, 05 Jul 2024	Prob (F-statistic):	0.0168
Time:	16:31:09	Log-Likelihood:	-46315.
No. Observations:	10000	AIC:	9.264e+04
Df Residuals:	9994	BIC:	9.268e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	34.0576	0.498	68.413	0.000	33.082	35.033
HighBlood_Yes	-0.7436	0.505	-1.471	0.141	-1.734	0.247
Arthritis_Yes	0.9118	0.519	1.758	0.079	-0.105	1.929
BackPain_Yes	0.8536	0.505	1.690	0.091	-0.137	1.844
Reflux_esophagitis_Yes	0.7355	0.505	1.457	0.145	-0.254	1.725
Asthma_Yes	-1.0403	0.548	-1.898	0.058	-2.115	0.034

Omnibus:	49454.969	Durbin-Watson:	0.363
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1079.320
Skew:	0.075	Prob(JB):	4.25e-235
Kurtosis:	1.398	Cond. No.	3.75

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

The equation for this model is:
Initial_days= 34.05 -0.74*HighBlood_Yes + 0.91*Arthritis + 0.83*BackPain_Yes + 0.74*Reflux_esophagitis_Yes – 1.04*Asthma_Yes

E. Analysis

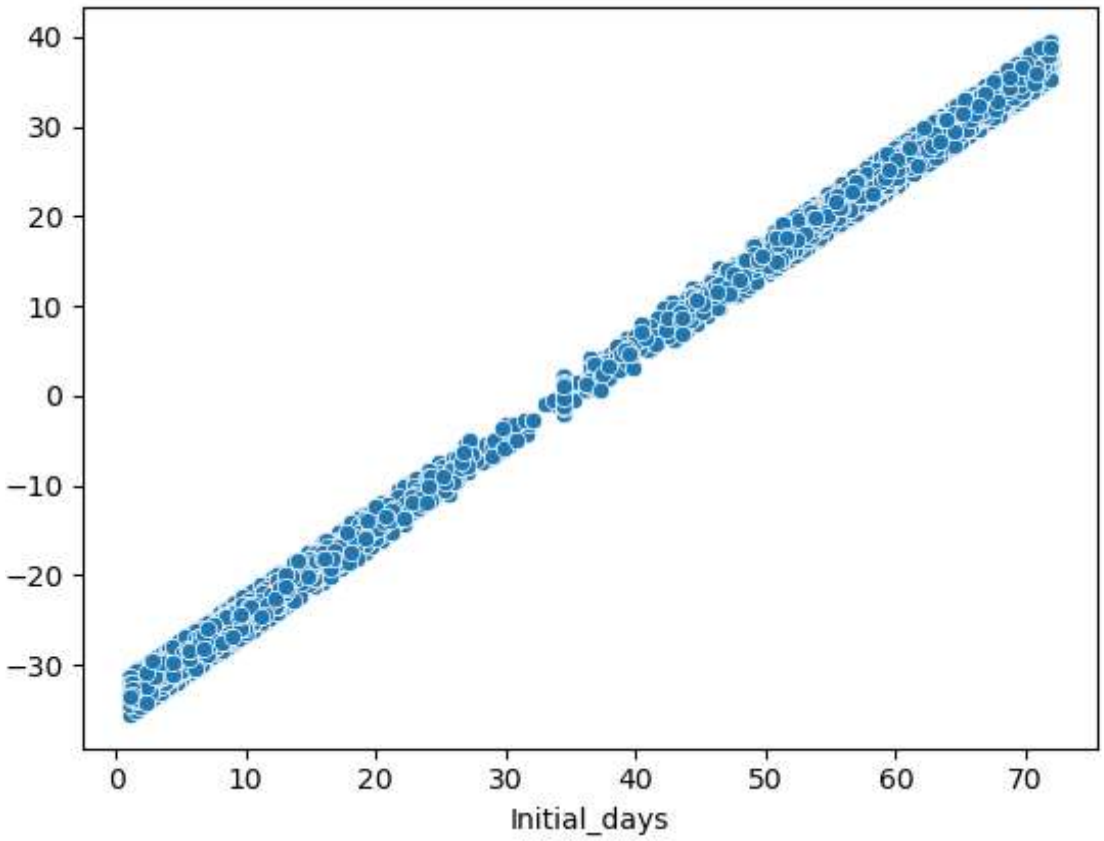
E.1.Data Analysis Process

The reduced multiple linear regression model can be compared to the initial model by comparison of the r-squared values. Unfortunately, in this case, neither of the R-squared values are high. The initial R-squared value is 0.002 and the reduced value is 0.001. The reduced model represents the data as well as the initial model. In this case we could use the retained values from the reduced model as a starting point to evaluate other aspects of the data for a better coorelation with Initial_days.

E.2. Calculations

See above for all calculations.

```
#Residual Plot
residuals = medical_df['Initial_days'] - lm_days_red.predict(medical_df[['intercept',
'HighBlood_Yes', 'Arthritis_Yes', 'BackPain_Yes','Reflux_esophagitis_Yes', 'Asthma_Yes']])

sns.scatterplot(x=medical_df['Initial_days'], y=residuals)
```



E.3. Python Code
See above for all code.

Part V
F. Summary

F.1. Discussion
The regression equation is:
Initial_days= 34.05 -0.74*HighBlood_Yes + 0.91*Arthritis + 0.83*BackPain_Yes + 0.74*Reflux_esophagitis_Yes – 1.04*Asthma_Yes
Interpretation of Coefficients: According to this model, if a patient has high blood pressure or asthma, their stay will be reduced. Having arthritis, back pain and acid reflux increased the patient's stay.
Significance: The R-squared value of this model is 0.001, meaning that the model only account for about 0.1% of the variation in the data. Overall this is a poor model. As you can see from the above equation, this model only accounts for about a 2 days difference from the average stay. However, it can be used as a starting point to re-evaluate the data set to look for other factors that correlate with length of stay.
Limitations: There are significant limitations to this model. To start with, this model is limited by the four assumptions discussed above. As mentioned above, the relationship between variable is assumed to be linear, however this may not be the case. An exponential or logarithmic equation may fit the data better. It was also assumed that the selected case were random, however, without knowing more of the data collection process, it is impossible to say that the data represents a random sample. The data might only be from a unit that specializes in cancer care, for example, the length 0of stay would be far more likely impacted by type of treatment or complications such as infection rather than the complications in

our model. Of course another limitation that should be mentioned, is that we are reliant on the accuracy of those who entered the data, which may not always be perfect.

F.2. Recommendations

From this model my recommendation would be to look at other aspects of the data set for correlation with length of stay. The question was can linear regression modeling be used to demonstrate correlation between Initial_days and any number of the per-existing conditions from the data set and the answer is no. Linear regression modeling demonstrates little to no correlation between length of stay (Initial_days) and per-existing conditions. I would recommend that the search for coorelation be expanded to all aspects of this data set to determine if any of the gathered data is correlated with length of stay. If there is no correlation then I would further recommend that a deep dive be performed on a few random cases with long lengths of stay to determine what the cause of the prolonged stay was. Once this is determined fro a few cases, new data should be gathered, including the data points that caused prolonged stays on the selected patients. This new data set should then be reviewed for correlation with Initial_days.

Part VI

G. Panopto video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=108f6618-f3f0-4b7b-a327-b1a5017d964c

G. Code Sources

"One Hot Encoding in Machine Learning." *GeeksforGeeks*, 21 Mar. 2024, www.geeksforgeeks.org/ml-one-hot-encoding/. Accessed Jan. 7AD.

Kumar, Ashwini. "A Quick Guide to Bivariate Analysis in Python." *Analytics Vidhya*, 18 Feb. 2022, www.analyticsvidhya.com/blog/2022/02/a-quick-guide-to-bivariate-analysis-in-python/.

"Detecting Multicollinearity with VIF - Python." *GeeksforGeeks*, 14 Aug. 2020, www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/.

H. Citations

"Pandas.DataFrame.drop — Pandas 1.3.2 Documentation." *Pandas.pydata.org*, pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.html.

"Detect and Remove the Outliers Using Python." *GeeksforGeeks*, 24 Jan. 2024, www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/. Accessed Jan. 7AD.

Sewell, Dr. William. "D208 Predictive Modeling Webinar Episodes 1-6." *WGU.edu*, wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=9c39f90e-4457-4613-b4cf-b109004601ed. Accessed 4 July 2024.

I. Professionalism

This paper endeavors to be professional.