**Part I: Research Question**

1. Describe the purpose of this data analysis by doing the following:
2. Can we predict the length of a patient's initial stay based on the patient information gathered in the data set provided?
3. Task One of Exploratory Data Analysis led to the discovery of a strong positive relationship between a patient's likelihood to be readmitted, represented by the variable ReAdmis, and the length of their initial stay. The goal of this multiple regression analysis is to predict the length of a patient's initial stay. This information would be useful for predicting a patient's readmittance probability with the logistic regression model found in Task Two.

**Part II: Method Justification**

1. Describe multiple regression methods by doing the following:
2. Multiple regression models assume that the independent and dependent variables have a linear relationship, that the residuals of the regression are normally distributed, that there is no multicollinearity between independent variables, and that the variance of error is equally distributed. The final assumption is often called the assumption of homoscedasticity.
3. Python, especially when coupled with Jupyter Notebook, provides a flexible development environment in which we can load libraries to solve complex problems with ease and minimal commands. The Pandas library allows Python to neatly organize data into data frames which can be used with many statistical and machine learning libraries. Numpy provides tools for calculating summary statistics and working with multiple slices of data frame objects at once. Seaborn, and the library upon which it is built, Matplotlib, allow for beautiful visualizations in concentrated commands. Sklearn, better known as Sci-Kit Learn, is a premier machine learning library containing methods for data preprocessing, model building, model evaluation, and reporting all in one place. Finally, statsmodels is a library containing useful modeling methods, like those found in sklearn, with an emphasis on ease of reporting model statistics.
4. Multiple regression is a proper model to answer the question found in Part A.1 because the outcome variable is continuous, not categorical, and depends on many factors. Multiple regression excels at predicting specific outcomes, rather than categorization, and is robust enough to handle numerous independent variables.

**Part III: Data Preparation**

1. Summarize the data preparation process for multiple regression analysis by doing the following:
2. To properly prepare the data, six steps are required. First, the data must be separated into four data frames depending on the preparation required for each variable: numeric variables must be grouped into one data frame, binary categorical variables into another data frame, non-binary categorical variables into another data frame, and ordinal categorical variables into a final data frame.

We must analyze the numerical data frame to identify outliers by computing the z-score of each value within each column and removing those with a z-score higher than three or lower than negative three; these values are more than three standard deviations from the mean and will be classified as outliers. We will drop the entirety of an observation when one or more of its values is declared as an outlier.

Using the pandas get\_dummies method, the non-binary categorical variables will be expanded into multiple binary variables indicating the existence, or lack of, a value for each unique value in the variable. The parameter drop\_first will remove one of the value-variables being created as a value of zero in each of the other newly created variables implies a value of one in the final option. This process is often called one-hot encoding and results in n-1 variables for each non-binary category variable where n is the number of unique values for that variable. Next, the binary categorical variables will be encoded with values of "Yes" being encoded as one and "No" as zero. The ordinal categorical variables are manually encoded, assigning lower values to lower risk values or population densities and higher values to higher risk values or population densities. Because there are only three ordinal variables in our data set, this encoding was easily constructed as a dictionary and applied through the pandas replace method.

To avoid bias in the model, we will normalize the numerical variables in the data set using sklearn’s MinMaxScaler. Finally, once these transformations are complete, we reunite the data as one data frame.

1. Our initial model will include the variable Initial\_days as the outcome variable and all other available variables, excluding CaseOrder, Customer\_id, Interaction, UID, City, State, County, Zip, TimeZone, and Job, as the predictor variables. The variables CaseOrder, Customer\_id, Interaction, and UID are excluded because they are unique identifiers that will not contribute to the model’s effectiveness. City, State, County, Zip, and TimeZone are excluded as they are all variables used to determine a patient’s geographic location, information which Lat and Lng provide in a more specific format. Job is excluded from the analysis due to the poor distribution of values across the categories. The highest count value in the Jobs variable has only 29 observations, representing only 0.3% of the observations in the data set.

Performing the describe method on the complete, unprepared data set yields the following summary:

CaseOrder Zip Lat Lng Population

count 10000.00000 10000.000000 10000.000000 10000.000000 10000.000000

mean 5000.50000 50159.323900 38.751099 -91.243080 9965.253800

std 2886.89568 27469.588208 5.403085 15.205998 14824.758614

min 1.00000 610.000000 17.967190 -174.209700 0.000000

25% 2500.75000 27592.000000 35.255120 -97.352982 694.750000

50% 5000.50000 50207.000000 39.419355 -88.397230 2769.000000

75% 7500.25000 72411.750000 42.044175 -80.438050 13945.000000

max 10000.00000 99929.000000 70.560990 -65.290170 122814.000000

Children Age Income VitD\_levels Doc\_visits

count 10000.000000 10000.000000 10000.000000 10000.000000 10000.000000

mean 2.097200 53.511700 40490.495160 17.964262 5.012200

std 2.163659 20.638538 28521.153293 2.017231 1.045734

min 0.000000 18.000000 154.080000 9.806483 1.000000

25% 0.000000 36.000000 19598.775000 16.626439 4.000000

50% 1.000000 53.000000 33768.420000 17.951122 5.000000

75% 3.000000 71.000000 54296.402500 19.347963 6.000000

max 10.000000 89.000000 207249.100000 26.394449 9.000000

Full\_meals\_eaten vitD\_supp Initial\_days TotalCharge

count 10000.000000 10000.000000 10000.000000 10000.000000

mean 1.001400 0.398900 34.455299 5312.172769

std 1.008117 0.628505 26.309341 2180.393838

min 0.000000 0.000000 1.001981 1938.312067

25% 0.000000 0.000000 7.896215 3179.374015

50% 1.000000 0.000000 35.836244 5213.952000

75% 2.000000 1.000000 61.161020 7459.699750

max 7.000000 5.000000 71.981490 9180.728000

Additional\_charges Item1 Item2 Item3

count 10000.000000 10000.000000 10000.000000 10000.000000

mean 12934.528587 3.518800 3.506700 3.511100

std 6542.601544 1.031966 1.034825 1.032755

min 3125.703000 1.000000 1.000000 1.000000

25% 7986.487755 3.000000 3.000000 3.000000

50% 11573.977735 4.000000 3.000000 4.000000

75% 15626.490000 4.000000 4.000000 4.000000

max 30566.070000 8.000000 7.000000 8.000000

Item4 Item5 Item6 Item7 Item8

count 10000.000000 10000.000000 10000.000000 10000.000000 10000.000000

mean 3.515100 3.496900 3.522500 3.494000 3.509700

std 1.036282 1.030192 1.032376 1.021405 1.042312

min 1.000000 1.000000 1.000000 1.000000 1.000000

25% 3.000000 3.000000 3.000000 3.000000 3.000000

50% 4.000000 3.000000 4.000000 3.000000 3.000000

75% 4.000000 4.000000 4.000000 4.000000 4.000000

max 7.000000 7.000000 7.000000 7.000000 7.000000

1. As outlined in Part C.1, the data must be divided into four data frames, one for numeric variables, another for non-binary categorical variables, one for binary categorical variables, and another for ordinal categorical variables. The numeric variables will have their outliers defined and offending observations dropped from the data set and the remaining values normalized with sklearn’s MinMaxScaler. The non-binary categorical variables will be one-hot encoded using the pandas library’s get\_dummies method. The binary categorical variables will be encoded manually, such that values of "Yes" become one and values of "No" become zero. The ordinal categorical variables will be manually encoded so that values representing lesser risk or population density are given lower values while values indicating higher risk or population density are given higher values. All these values are scaled between zero and one to ensure that the model is not biased to extra range in any variable. Finally, these four data frames are reunited. The code used to generate these transformations is provided below:

#Divide the data depending on the variable type

data\_num = data.iloc[:, np.r\_[8:11, 14:17, 20:24, 39:50]].copy() #Numeric variables

data\_enc = data.iloc[:, np.r\_[17:19, 38]].copy() #Non-binary Categorical Variables

data\_yn = data.iloc[:, np.r\_[19, 24, 26, 27, 29:38]].copy() #Binary Categorical Variables

data\_ord = data.iloc[:, [11, 25, 28]].copy() #Ordinal Categorical Variables

def meddata\_preprocessing(data\_num, data\_enc, data\_yn, data\_ord): #This is saved as a function for reusability in later tasks

for i in range(len(data\_num.columns)):

#Compute the mean and standard deviation of each column

mean, std = np.mean(data\_num.iloc[:,i]), np.std(data\_num.iloc[:,i])

#Set the upper and lower bounds at three standard deviations from the mean

upper, lower = mean + 3 \* std, mean - 3 \* std

#Record the index for each row that contains a value outside the previously set boundaries

drop = [inx for inx, x in enumerate(data\_num.iloc[:, i]) if x < lower or x > upper]

#Compare the list of indices to be dropped with those within the dataframe and drop those that remain

#within the data frame while ignoring those that were already dropped

for d in drop:

if d in data\_num.index:

data\_num = data\_num.drop(d)

#One-hot encode categorical variables

data\_enc = pd.get\_dummies(data\_enc, prefix=data\_enc.columns, drop\_first=True)

for col in range(len(data\_yn.columns)):

#Replace values of "Yes" with 1 and values of "No" with 0

for inx, val in enumerate(data\_yn.iloc[:, col]):

if val == 'Yes':

data\_yn.iloc[inx, col] = 1

else:

data\_yn.iloc[inx, col] = 0

#Determine levels of ordinal variables

scale\_mapper = {

"Area" : {

"Rural" : 0,

"Suburban" : 0.5,

"Urban" : 1

},

"Initial\_admin" : {

"Emergency Admission" : 1,

"Observation Admission" : 0.5,

"Elective Admission" : 0

},

"Complication\_risk" : {

"Low" : 0,

"Medium" : 0.5,

"High" : 1

}

}

#Replace values with numerical equivalents specified above

for col in data\_ord.columns:

data\_ord[col] = data\_ord[col].copy().replace(scale\_mapper[col])

mm = MinMaxScaler() #Instantiate the MinMaxScaler method

data\_num[data\_num.columns] = mm.fit\_transform(data\_num) #Standardize the data

#The use of an inner join preserves the dropping of rows performed on data\_num

#The data\_yn dataframe is converted to a numeric datatype, int32, before joining

data\_clean = data\_num.copy().join(data\_enc, how='inner').join(data\_yn.astype('int32'), how='inner').join(data\_ord, how='inner')

return data\_clean #Returns a fully prepared data set

#Call the preparation function

data\_clean = meddata\_preprocessing(data\_num, data\_enc, data\_yn, data\_ord)

data\_clean #Preview the data

A copy of the Jupyter Notebook file containing this code is also attached as an .ipynb file as well as an HTML file.

1. Univariate distributions:

#Univariate distributions of all independent variables

fig, ax = plt.subplots(14, 3, sharey=False, figsize=(14, 64))

axi = 0

for i in np.r\_[8:12, 14:50]:

ax.flatten()[axi].hist(data.iloc[:, i])

ax.flatten()[axi].set\_title(data.columns[i])

axi += 1

Chart, histogram

Description automatically generated

Chart, waterfall chart

Description automatically generated

Graphical user interface

Description automatically generated

Chart, waterfall chart

Description automatically generatedChart, histogram

Description automatically generated

Bivariate:

#Bivariate distributions for independent variable vs. the dependent variable

fig, ax = plt.subplots(8, 3, sharey=False, figsize=(16,32))

inx = 0

for i in np.r\_[8:12, 14:17, 20:24, 39:50]:

ax.flatten()[inx].scatter(data.iloc[:, i], data['Initial\_days'], )

ax.flatten()[inx].set\_title(data.columns[i])

ax.flatten()[inx].set\_ylabel('Initial\_days')

inx += 1

A picture containing diagram

Description automatically generated

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated with low confidence

#Bivariate distributions continued

fig, ax = plt.subplots(7, 3, sharey=False, figsize=(14,32))

axi = 0

for i in np.r\_[11, 17, 18, 19, 24, 25:39]:

sns.boxplot(data=data, x=data.columns[i], y='Initial\_days', ax=ax.flatten()[axi])

ax.flatten()[axi].xaxis.set\_ticklabels([])

axi +=1

Chart, treemap chart

Description automatically generated

A picture containing square

Description automatically generatedChart, bar chart, box and whisker chart

Description automatically generated

1. A copy of the prepared data is provided as med\_data\_prepared.csv.

**Part IV: Model Comparison and Analysis**

1. Compare an initial and a reduced multiple regression model by doing the following:
2. The initial model is created by separating the target variable, Initial\_days, from the predictor variables. We place the predictor variables in the data frame X and the target variable in the series y. Statsmodels’ add\_constant method allows us to provide a constant to the model. The statsmodels OLS method is then used to generate a linear model which is fitted to the target and predictor variables, from which we can examine the significance of each variable and identify variables that are statistically insignificant.

Input:

#Divide prepared data into predictors (X) and target (y)

X = data\_clean.drop('Initial\_days', axis=1).copy()

y = data\_clean['Initial\_days'].copy()

#Add constant

X = add\_constant(X)

#Instantiate and fit linear model

lin\_reg = sm.OLS(y, X)

model\_init = lin\_reg.fit()

model\_init.summary()

Output:

Graphical user interface, application, table

Description automatically generated with medium confidence

Table

Description automatically generated with medium confidenceGraphical user interface, table

Description automatically generated

The output shows us that this model predicts the length of initial stay almost perfectly, with an adjusted R-squared value of 0.995. Despite the apparent success of the model, there is much that could be done to tidy the model. Primarily, the model has not been verified to satisfy the assumption that the independent variables are not correlated, the assumption of non-multicollinearity; additionally, we can see that the p-values of many of the variables are well above the traditional level of statistical significance, 0.05.

1. We employed two methods of feature selection in the refinement of the multiple regression model. First, the assumption of non-multicollinearity was used to remove any independent variable that had a strong correlation with other independent variables. Collinearity between independent variables was identified with a heat map representing the correlation matrix of the data frame data\_clean from which the first model's X data frame was created.

Chart

Description automatically generated

In the heat-map, we can see that the variable Zip has a strong negative correlation with Lng and TimeZone, Age and Additional\_charges are highly correlated, the ReAdmis variable is highly correlated with Initial\_days and TotalCharge, the Item variables all have correlations with one another, and HighBlood and Additional\_charges are highly correlated. To remove this multicollinearity, the Item variables and Additional\_charges are dropped from the data frame X for the creation of the second model.

Now that the assumption of non-multicollinearity has been satisfied, a second method of reducing the initial model is employed. Statsmodels' summary method provides many data points concerning the multiple regression model. The output we will focus on is the p-value of each independent variable, found in the column P>|t| in the output found in Part D.2. A high p-value implies that a variable is not statistically significant (JHSPH p.125). For this model refinement process, we will set the threshold of significance at 95%, requiring all variables to have a p-value below 0.05 to be considered significant. Once the highest p-value has been identified in a model's output, if it is above the significance level 0.05, we will drop it from the independent variable list and the model created anew. We will repeat this process until all independent variables have a p-value below the specified significance level.

1. The model is reduced through the application of the following function, which identifies and drops the variable with the highest p-value if that p-value is above the threshold of statistical significance, 0.05, and returns a reduced model.

Input:

#Model reduction strategy is saved within a function for reusability

def reduce\_model(X, y, sig\_lvl):#Accepts predictor df, target series, and significance level

lin\_reg = sm.OLS(y, X)#Creates model

model = lin\_reg.fit()#Fits model

p\_val = model.pvalues.to\_list()

var = X.columns.to\_list()

while max(p\_val) >= sig\_lvl:#Breaks once all predictor variables are statistically significant

for i in range(len(var)):

if p\_val[i] == max(p\_val):

X.drop(var[i], axis=1, inplace=True)#Drops variable with highest, non-significant p-value

lin\_reg = sm.OLS(y, X)#Recreates model

model = lin\_reg.fit()#Refits model

p\_val = model.pvalues.to\_list()

var = X.columns.to\_list()

return model, X #Returns a reduced model and the corresponding predictor variables

Output:

Table

Description automatically generated

Table

Description automatically generated

We can see in the output that all the variables have p-values less than 0.05 and can be deemed statistically significant. Another observation from this output is that the adjusted R-squared value remains 0.995, indicating no loss or gain in goodness-of-fit from the initial model to the reduced. This is likely due to the significant correlation between the independent variable TotalCharge and the dependent variable Initial\_days.

It is worth noting that through all revisions of the model development, these adjust R-squared values were never different. This implies that the similarity in adjusted R-squared is not an artifact of the process or preprocessing but is derived from the heavy correlation between the target variable, Initial\_days, and one predictor variable, TotalCharge.

1. Analyze the data set using your reduced multiple regression model by doing the following:
   1. We accomplished feature selection by comparing the maximum p-value for the model to the level of statistical significance, 0.05 for our purposes. If the maximum value is above the significance level, the variable with that p-value is dropped from the predictor data frame and the model is re-instantiated and refit to the new, reduced data. This check is repeated until the maximum p-value remaining is less than the threshold of statistical significance. Our reduced model contains only 25 predictor variables. By comparing the initial data frame containing all variables, X, with the reduced data frame, X\_final, we can see that the following variables have been removed:

Input:

print([x for x in X.columns.to\_list() if x not in X\_final.columns.to\_list()])

Output:

['Lat', 'Lng', 'Population', 'Children', 'Age', 'Income', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten', 'Additional\_charges', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8', 'Marital\_Married', 'Marital\_Never Married', 'Marital\_Separated', 'Marital\_Widowed', 'Gender\_Male', 'Gender\_Nonbinary', 'Services\_CT Scan', 'Services\_Intravenous', 'Services\_MRI', 'Soft\_drink', 'Stroke', 'Overweight', 'Asthma', 'Area']

We compare the effectiveness of each model by comparing the adjusted R-squared metric of each model provided by the summary method for the initial and reduced model. As discussed briefly in Parts D.1 and D.3 above, the adjusted R-squared for both the initial and reduced models is 0.995. The chief advantage of the reduced model over the initial model is not the goodness-of-fit, represented by the adjusted R-squared, but the increased simplicity.

An adjusted R-squared value this high is indicative that the model is likely over-fitting the training data and may not translate well to test data. This may be explained by the over-reliance on TotalCharge, but the possibility of over-fitting is important to acknowledge.

A likely culprit for the lack of difference in the adjusted R-squared of the two models is the variable TotalCharge. Because this variable is highly correlated with the target variable and is present in both models, it is likely that this variable accounts for a large portion of the variance in the target variable while a small amount of the variance is derived from differences in the remaining variables in the reduced model. The variables removed from the initial model to arrive at the reduced model appear to have no impact on the target variable.

Comparing the residual plot of the initial model with the reduced model shows a notable improvement through model reduction. The initial model's residual plot appears to have six main groupings and ranges from -0.05 to 0.06; the reduced model appears to have further divided the groups from the initial plot into distinct, linear groups, ranging from -0.05 to 0.06. The presentation of the residuals in the final model implies that the dataset satisfies the assumption of homoscedasticity as there is little linear relation shown in the plot.

* 1. Residual Error of reduced model: 0.025553433522022162

Reduced model coefficients:

const 0.044076

vitD\_supp -0.004455

TotalCharge 1.219207

ReAdmis 0.011825

HighBlood -0.018208

Arthritis -0.011372

Diabetes -0.012685

Hyperlipidemia -0.015740

BackPain -0.014182

Anxiety -0.014624

Allergic\_rhinitis -0.011838

Reflux\_esophagitis -0.010401

Initial\_admin -0.094657

Complication\_risk -0.078014

Predictions of reduced model:

[0.12613338464055443,

0.21927295101181093,

0.054652869279365905,

0.012055214546240263,

0.040984064925655206,

0.022313022202834237,

0.145422427083934,

0.13734053072597663,

0.10431619032578941,

0.030103772751716523,

0.01622533768548181,

0.11553025987344741,

0.027033763970275286,

0.03814249331510844,

-0.007915816135914834,

0.0075858914927466395,

0.0006707262622600435,

0.08790675731768172,

0.15825586430760197,

0.1072556782612434,

0.06257420252164698,

0.13975908750672553,

0.2780356412485319,

0.10610372299287281,

0.08310542687458086,

0.2595596734820635,

0.20505437624265582,

0.15104845764162392,

0.1038113236783201,

0.3772445162441006,

0.1034502120438252,

0.1359490988719114,

0.11591914315421853,

0.02695404000308905,

0.13861216979254865,

0.1661822135127576,

0.12433123893118478,

0.058746174329487075,

0.09634903118904968,

0.1478650280742363,

0.16576039991810718,

0.2212077977069116,

0.13703767986341253,

0.1527762185815643,

0.0195925613711058,

0.20788154355767263,

0.19475033384311313,

0.2324648585952615,

0.10905481042103615,

0.018889387245348548,

0.04069871216752968,

0.005136705167258941,

0.05672207517157945,

0.16678518743827925,

0.20829052191364256,

0.30712686486307095,

0.059605544564782634,

0.295418405707319,

0.007145602349493289,

0.10696117011748627,

0.4313794935204051,

0.1818579912016206,

0.13646389835360176,

0.06583986952604338,

0.040104816493175884,

0.24588237472935004,

0.2636739155231857,

0.09665978189697588,

0.0639769321572063,

0.060429329053655736,

0.12389948181963836,

0.1270897052924731,

0.10754783950693111,

0.07289043062640754,

0.15054572568697883,

0.10054903203682818,

0.27114207868969614,

0.07658064791803858,

0.15181169409185186,

0.06574897225409496,

0.08824010661286472,

0.020448342048405954,

0.15113909413237728,

0.13862322892610202,

0.046613871931747396,

0.1762672354552995,

0.012906503750690072,

0.03912290470596455,

0.025776837805315922,

0.11879248353214518,

0.04689543038404188,

0.04081006372912219,

0.07484119202688097,

0.3118171985623692,

0.04025732997261146,

0.1322653611618324,

0.3125503665694685,

0.08298565548540393,

0.0714160449678435,

0.10160324752889979,

0.08244595484295385,

0.09404794329653478,

0.10838465724971308,

0.04053514872593098,

0.09512663652784391,

0.055392651588736165,

0.07882543648520376,

0.08893789360054008,

0.04724436402996544,

0.07079166338360389,

0.16202744985532083,

0.014315209927851832,

0.2050084145102557,

0.12386493231162268,

0.11691589652678337,

0.12427412501772864,

-0.02308505011312681,

0.005049954315992693,

0.13565676176436717,

0.05242256876689697,

0.09409922687569637,

0.06655228089801535,

0.08405889558754306,

0.14030103534192503,

0.07236373124921598,

0.05164399304113489,

0.1392032324101346,

0.017033722283039975,

0.02801983759204007,

0.0026741246331752136,

0.04829181293998504,

0.22756840263342115,

0.15330685119711926,

0.0527902915369853,

0.17272006717413288,

0.046350383939927305,

0.09267108812927534,

0.130151132009364,

0.2726172246696495,

0.3940064561700501,

0.11653115995371444,

0.26665319125799264,

0.012515415861066484,

0.08326436814019912,

-0.006797207637354474,

0.17855109890175158,

0.18867898347409978,

0.10187164014178887,

0.100313087522855,

0.1734758085349459,

0.18146879542666522,

0.21490183676450464,

0.2163482950013984,

0.22634371694759392,

0.04698592060886237,

0.0004096536157767311,

-0.0027466867658300156,

0.2097199251775197,

0.24939618946545142,

0.07229001375518448,

-0.0047426971721199335,

0.27016438569963974,

-0.006965513294094235,

0.05645480975686619,

0.12869437128700958,

0.1794598404431195,

0.22563228035247468,

0.18192407605440225,

0.013498929625630367,

0.06964574556154698,

-0.00624772877929345,

0.06567251971822191,

0.017566805026090586,

-0.010426307607835084,

0.10664560255001837,

0.12594454034930047,

0.2928219864540966,

0.12170781378990511,

0.14916675837829518,

0.00012482529130974224,

0.13820638523215995,

0.14773544588322504,

0.03092057078264169,

0.017637436043691453,

0.11003798182981728,

0.18519418042481195,

0.11919946201639993,

0.17506739261029683,

0.06864300960994232,

0.22485996660700236,

0.2188555999329125,

0.0038504032519551187,

0.02850374893190255,

0.22350411716988428,

0.05122800851292243,

0.11496736889509668,

0.09668245628670702,

0.1225090042500988,

0.2778794592534944,

0.10672220041013049,

0.12050744386934628,

0.15258242584647455,

0.12129306544378296,

0.2315011430990112,

0.03601305287720228,

0.0875851717376967,

0.111110777251342,

0.12423891422947635,

0.15044627171775068,

0.031962798354726846,

0.14017718010526878,

0.07624405105591466,

0.008739762553462989,

0.12128082168893246,

0.12656854606349258,

0.1185245029910988,

0.1064764370502704,

-0.022084089354648576,

0.058995163257820234,

0.10814374264267729,

0.26500917063744245,

0.1515742572398421,

0.17664643122266163,

0.16749953724674801,

0.07617133961159535,

0.24119487262353817,

0.11643270077840641,

0.0769475352007753,

0.10073400776243939,

0.1886677980732046,

0.026399897847150383,

0.1619938282671142,

0.36171558414495947,

0.11024651294847485,

0.096241041424205,

0.07794286069795,

0.008883016271607168,

0.1558075225527677,

0.20593966905609368,

0.15953662942289085,

0.07935588661478041,

0.18976271373936682,

0.06668127060738356,

0.3418352495020931,

0.03070839912493164,

0.1419509386948018,

0.2453223646158041,

0.043428063242724066,

0.06770292982304593,

0.09473624025212873,

0.017271595514909435,

0.3176353459410247,

0.14073476253340816,

0.13365159750044714,

0.004998727674184571,

0.17587466883535763,

0.08164685664642574,

0.014240768075349457,

-0.01353674418051809,

0.24764623934978947,

0.10333125711567873,

0.3408279317997299,

0.036314630761915885,

0.09984160134139104,

0.1918885247170526,

0.05733930623118495,

0.11216069633790914,

0.07088176941696067,

0.18883374158464009,

0.19520756456645216,

0.2773207795929751,

0.12130214071873054,

0.3016075360995363,

0.07996213665566289,

0.024887688636418208,

0.041112038387916876,

0.1441406138666841,

0.10492085484767748,

0.21090357740659071,

0.14888848280945427,

0.0742732741200555,

0.12452820630331475,

0.013045216950875863,

0.21254600256539335,

0.044965382291301925,

0.28135900617547793,

0.027150454278332452,

0.15963636894475228,

0.15800064723617235,

0.0435334163422718,

0.30998073765842016,

0.09666059168069782,

0.019106026483030683,

0.2057554524725928,

0.3986663192329072,

0.17352148927638233,

0.3385649990981411,

0.1054951934573026,

0.07348267991578264,

0.2692668044809729,

0.05936906114748138,

0.27823538307917794,

0.029169886121045596,

0.18663492761878417,

0.1115803612293,

0.23914583095466263,

0.2361353587712773,

-0.0020879733833386177,

0.06554701483555687,

0.05483772572180513,

0.005330279205498681,

0.14575256388567187,

0.11491633057170948,

0.08506176782876726,

0.056051455445332826,

0.3005112648986637,

0.0383719249179808,

0.13191488467843282,

0.1715466916269915,

0.03611880145567408,

0.00313852045305877,

0.13066662388307704,

0.09905791772095043,

0.07500658953090338,

0.17173770194933727,

0.10372553571510107,

0.19089607126589253,

0.09092193694686883,

0.08465710153087888,

0.20401384254440472,

0.11395248705419737,

0.09864214093023457,

0.13157714416131694,

0.09114098293259884,

0.02804641191020142,

0.11402293789156684,

0.046726907022685824,

0.058650598148151836,

0.041064418066458504,

0.030528167161371994,

0.05703540089818872,

0.2339668550305561,

0.10942641442911233,

0.13159040249443482,

0.11920880925157397,

0.05345257694891011,

0.15710857308784537,

0.09594188056827219,

0.017039175866757926,

0.051488927125541324,

0.02741211384352763,

0.13936148550187044,

0.1800252710337644,

0.32914447654299944,

0.08916662360820254,

0.01857629001017775,

0.08590585397366071,

0.016578603919390936,

0.10861414433978603,

0.2729035500478064,

0.10661380788693964,

0.07666817641519329,

0.07445004768202208,

0.24803350539885655,

0.055193187053332246,

0.03275760241946451,

0.1435423306298859,

0.03229525663076206,

0.11479239589888154,

0.21277520874476527,

0.20203820218495222,

0.19416606747553689,

0.2135872606191408,

0.06124094104062742,

0.09752568984212512,

0.19369630820924086,

0.04991992489895834,

0.09161520918583599,

0.05780604472789773,

0.09303318342271556,

0.1667171551067777,

0.2309063133341129,

0.09107182636847475,

-0.028119629381776562,

0.06922199178576359,

0.13866318277274486,

0.004366674437086063,

0.1045060644567356,

0.021819723947592233,

0.08719485055100525,

0.08626119245673078,

0.23285054815252065,

0.08911824372260053,

0.04376649190721446,

0.22266572706861457,

0.029226121898829845,

0.15708821452541705,

0.1555800031544184,

0.2968127085455888,

0.059176712914628136,

0.050139019884352085,

0.12331568342810124,

0.11880827364442817,

0.059855210048788035,

0.09863128194377237,

-0.03313718780873194,

0.17077734662159563,

0.08600275806061597,

0.19463853917823484,

0.10459704104228026,

0.14968512454188782,

0.20882835470921302,

0.1101405669941061,

0.18527854136419347,

0.2288723519754712,

0.13117803165682881,

0.28101533784124977,

0.0800500078854856,

0.2082441632325942,

0.12414847088002143,

0.2124995784621468,

0.09512549317053334,

0.05498391267511789,

0.059475412483026194,

0.06699369798820234,

0.049811040208106803,

0.14111006892650096,

0.16814615155006568,

0.09634152329449681,

0.2018335353504063,

0.29903417162048473,

0.015373991938534746,

-0.0012555575108473618,

0.09292617667114145,

0.07198089499429072,

0.050868533656959355,

0.03920402221390545,

0.008413304087114597,

0.08290740098065294,

0.1817743086346289,

0.22516979075722604,

0.06993084372752241,

0.09454410947240409,

0.22343714241042203,

0.08372035535684288,

0.12843633886830122,

0.1128866599855332,

0.05013408395600118,

0.004830439462601893,

0.18762947932089663,

0.334622993276343,

0.027959141864396145,

0.1358969620185923,

0.3319536272404779,

0.015796294055937146,

0.31060557153204954,

0.033210518175287626,

-0.011945956056425802,

0.1796208015066952,

0.20875902289775855,

0.09828103925851012,

0.22120858250514142,

0.09516817378147635,

0.11187067672080005,

0.325965694070288,

0.07548246313222473,

0.07597835710781009,

0.12122027630918605,

0.13772744763227468,

0.06956362194919226,

0.18707744998593917,

0.19618955161574353,

0.03808006209816925,

0.2829366052811061,

0.218093560565372,

0.17219679893315157,

0.10489215843131414,

-0.03467587928393433,

0.02770936476897992,

0.09863356506813176,

0.0352692686472237,

0.01582074524836767,

0.062082618919635395,

0.020319627452743176,

0.1974898040011426,

0.12367593147238867,

0.1370866843951372,

0.09593893658579146,

0.19782390018612978,

0.04329949271562315,

0.05984008251877472,

-0.017664760494029944,

0.018310964167455285,

0.23862352930813527,

0.0705893231652443,

0.03945782556329806,

0.3047039813491109,

0.17943472793130522,

0.11665372043660906,

0.03521714852511492,

0.014597779238883508,

0.25492828763366615,

0.441728114975737,

0.22705063915663623,

0.014874227369778635,

0.09034653694760914,

0.14654832447792576,

0.0856346848090884,

0.28450945844229525,

0.04595870681689606,

0.08956146862057163,

0.1861082509165371,

0.010079124816722691,

0.0566616823131556,

0.23159623708550103,

0.1635655439562018,

0.0693495789985148,

0.11085773133117628,

0.16588475666417282,

0.19260591421148007,

0.1354384098786126,

0.11229334681290855,

0.21898536045572042,

0.17570246818415056,

0.23891645877034767,

0.12433808956500154,

0.1391746867831739,

0.31001096134309497,

0.009239633201504636,

0.1795510162513867,

0.11007328954810879,

0.2597552326267196,

0.008607079145461535,

0.1909883533176993,

0.21597608989293055,

0.23347071496035188,

0.11624337720274058,

0.09609472085290034,

0.2287693682233,

0.008085742192842485,

0.027137123487491886,

0.1273235402393147,

0.16524645163025714,

0.07156566904446711,

0.1205037568220462,

0.07254506384069849,

0.10768624761639525,

0.02741747538804227,

0.14933171460243805,

0.20754629887801632,

0.196469293337047,

0.1303278840453391,

0.15947626428050996,

0.3711399643817488,

0.09363720144044088,

-0.040044925910634216,

0.053837518366768325,

0.19349654456943155,

0.034578958565018726,

0.12802636716085666,

0.12224130660681207,

0.171727963404887,

0.08663915565161176,

0.01144004720522162,

0.08566364047112526,

0.21539873106839985,

0.04605628411849138,

0.20550489981841072,

0.11385519039405054,

0.12066330396178727,

0.11110601845560089,

0.048254345753716135,

0.1357425724319909,

0.1992000789537139,

0.14428767286333355,

0.17913344408032483,

0.07535948308200956,

0.18217345399136484,

0.24810481287882413,

0.026317315589997052,

0.14837193546941257,

0.07486164949616946,

0.11595692552866273,

0.2670187978168048,

0.04837012163791276,

0.1292031384896127,

0.0290973232986921,

0.035090058243873894,

0.08501025842916102,

0.017652590354665397,

-0.002850038963359898,

0.12679056978859343,

0.15282012342698945,

0.09092439909459647,

0.22827798461327287,

0.06360884287407755,

0.10001974628440116,

0.09923543179622148,

0.04580512834555007,

0.044517465606047475,

0.22723610478737552,

0.4836933941583534,

0.21659684743525448,

0.21806667794972653,

0.22895670853064098,

0.05477000912242883,

0.045183736150898476,

0.10559459178534555,

0.060083459674954084,

0.162191203318098,

0.17669024825382582,

0.045600774720980834,

0.3288410640994077,

0.09145514331728612,

0.1864793250322283,

0.05779278937942255,

0.1878422451715558,

0.01068795684238303,

0.21992814388713222,

0.2153952889259813,

0.03333533513960185,

0.05562648385599818,

0.16589083875604138,

0.17873606178248153,

0.10933151966156875,

0.10580640628516715,

0.15060938345606145,

0.07803754658098418,

0.16166671675322208,

0.14583367906947098,

0.10295690360507721,

0.059496369019983994,

0.06113984916872546,

0.13973877912224028,

0.012850739647923902,

-0.008531502616455952,

0.06597580379033721,

0.06431460879727006,

0.06886517189343375,

0.3440193712261063,

0.10622652504027998,

0.258744082299013,

0.38651718580864797,

0.06194842661028195,

0.16246351857187807,

0.003298180797921095,

0.06558526669185223,

0.13443965220462945,

0.2446082915766175,

0.03719647752111662,

0.12026915060147,

0.15332059245504498,

0.0586199444347552,

0.07075993229259386,

0.13561303790236617,

0.40206608216692896,

0.20123171570334056,

-0.0021768615578547437,

0.15320391205796696,

0.14661482750541013,

0.32040367869904945,

0.04811221735578848,

0.10727944367294598,

0.197670116882876,

0.06388273604308048,

0.047852662312607534,

0.07797129895282841,

0.25114724282141404,

0.23797870244508865,

0.15848477114615936,

0.07097419793342932,

0.0835778355590033,

0.010366686445588899,

0.06806378065127612,

0.08095677616351032,

0.11440160950497359,

0.05814086056848912,

0.05239093307116112,

0.17942468624089375,

0.2193292729375411,

0.2665927180027053,

0.2203119118906981,

0.0869118509708702,

0.11869320538343604,

0.1259037123918645,

0.0737209287122087,

0.12725007316828385,

0.223517236031248,

0.08335709034607207,

0.03968309879703039,

0.1431116782783497,

0.0818210742081467,

0.06709402677903821,

0.1468912926784732,

0.29407574338152526,

0.03582234227743748,

0.012570643185668876,

0.06068693938556965,

0.063287547807023,

0.004485271034023276,

0.15513802756311376,

0.11161279761770307,

0.04018867748326778,

0.18123928719041732,

0.08830233455849651,

0.23684877008817956,

0.17578215230401667,

0.1490274676933409,

0.19255954973450312,

0.1820900482318582,

0.11914978542535723,

0.07420181458163454,

0.113254657926364,

0.2421912108238294,

0.02660475969981605,

0.2772122209502174,

0.1761986877248134,

0.13640572534455314,

0.047560428928194225,

0.15645509395023807,

0.038179669240103985,

0.07316769517866595,

-0.0025069764267978967,

0.020367683114624273,

0.07025830794297466,

0.05908088845181382,

0.12563254877149754,

0.040100393212291925,

0.1252965868047896,

0.09074380018690767,

0.020106273131984417,

0.09916028040254618,

0.28032513829161215,

0.07368482282064048,

0.14905232537112437,

0.1846234693226201,

0.1731852679574901,

0.04076477116738325,

0.16407921754534294,

0.0062975662132053675,

0.23927458959940154,

0.22767397719346258,

0.1584267430516698,

0.14846978523552504,

0.10085153882239507,

0.22338902532407953,

0.06160121986266955,

0.25802869197439876,

0.027647512315847805,

0.09889332655773295,

0.04547728609879901,

0.11893189383180225,

0.03144669083006312,

0.05940640509634439,

0.19553270150493354,

0.0094659723971088,

0.2762421248191331,

0.11517649168184667,

0.06853828955282378,

0.024423443700814555,

0.11983662089743503,

0.07449169414909837,

0.1516118471323354,

0.14380577078674248,

0.005636134899791277,

0.08240882361020199,

0.03687757542599755,

0.1489464730172789,

0.21277956074554,

0.06955335709628327,

0.18453360031056104,

0.19759747029633584,

0.10285725332824194,

0.3220699868962142,

0.07773818192889777,

0.03391539094513447,

-0.005805776983268236,

0.1466189887971388,

0.26798652764943065,

0.2184476079435135,

0.03588818353489326,

0.057833618287942104,

0.0030937445883659903,

0.1843630483427753,

0.06563000376312497,

0.0974569175347141,

0.03860682400975215,

0.11838870132903162,

0.03716193245296585,

0.2512518250321007,

0.060870195834032984,

0.06417042138937641,

0.04324560442783923,

0.03842261504758917,

0.239910743757698,

0.37541896004389746,

0.08824750946179483,

0.12309347695183481,

0.10790198448383388,

-0.00831492157808153,

0.17594650999175912,

0.12345426018973739,

0.08144595623644166,

0.008135565522780878,

0.061375281575547495,

0.05932026164064115,

0.10237471941133337,

0.15346657442265915,

0.09210506135773218,

0.08870427214370712,

0.11163622244265674,

0.14598350199625942,

0.0981301947490448,

0.05901589886294512,

0.18145170531691324,

0.0452994143827932,

0.15729764817091524,

0.05809355453850442,

0.1778195593460458,

0.11344825111390604,

0.07789449155296051,

0.24293800643130037,

0.018416520684321397,

0.0452225716751612,

0.22305895452389018,

0.10201436926464881,

0.12652036185733123,

0.08222418667764081,

0.10251852993866746,

0.1362899895270925,

0.18694568503444361,

0.2248755296431553,

0.10695422081321888,

0.057871479188127364,

0.05692349686815041,

0.09579552441651669,

0.06443153399501317,

0.10593338966768416,

0.25963576408772177,

0.09333559737196946,

0.14007193310943064,

0.19338205804343614,

0.08949338357977907,

0.1036667582136667,

0.09466167440452673,

0.0927273527026289,

0.10736844863882547,

0.07167819427005036,

0.12408409321176371,

0.13977453879817608,

-0.045940050939665855,

0.10900499484452934,

0.048227727327607425,

0.12257841349983482,

0.1422106751817081,

0.02846954808971309,

0.1550088513973029,

0.16146323397038126,

0.11721365775590407,

0.060596291747194185,

0.13020348994677758,

0.05465780757713397,

0.025561102167332282,

0.043263500752828546,

0.137350296121525,

0.04930860705400602,

0.1441145676121858,

0.0385636228593229,

0.10237500777205173,

0.09014263486854539,

0.15822401037780293,

0.06739141486093941,

0.08401167609090061,

0.04049081357157562,

0.04298980755051981,

0.0664430685571312,

0.14496492678102238,

0.08660807064992511,

0.12723041508725508,

0.19260248857548135,

0.05507254138640861,

0.17861584290696067,

0.0466967881885269,

0.0344984722197058,

0.4374156951903642,

0.1653916147889075,

0.21938866627304635,

0.05845048736827812,

0.5157729853488148,

-0.004441506800876717,

0.17264328762343037,

-0.0058907088144933145,

0.13617884169109618,

0.40744040575133167,

0.08914388866937148,

0.08251759809956603,

0.14258643815448305,

0.031321027325904506,

0.0033974863597640154,

0.1450869385534617,

0.24511606556475604,

0.015162861422030736,

0.20397421459314824,

0.080075982198142,

0.2908484420278763,

-0.0012743896437109464,

0.0036510375320666436,

0.13301699529128524,

0.03578509764748388,

0.05014403382333317,

0.08902817221416656,

0.061075692946218584,

0.03896974825391663,

0.07927370168514493,

0.05605841338345076,

0.07593704442737144,

0.2061281646850496,

0.18097953017598029,

0.025826786143866375,

0.18663445786297636,

0.030137583910435806,

0.17443401912456355,

0.06458208553253066,

0.10845576700934909,

0.2092009488748733,

0.04708549653620362,

0.04646694287787179,

0.04983175343046327,

0.09536927815558617,

0.017347396293858834,

0.1252095656356416,

0.09717565463658707,

0.11995364367304857,

0.34248795347166644,

0.26897088364933186,

0.2229243182038343,

0.1272849000248713,

0.10716081889863632,

0.1949857965936529,

0.03295062978813414,

0.38249794961269606,

0.09155663786070735,

0.04448000912271652,

0.1708486397768467,

0.10711946363380513,

0.13699465203445296,

0.0434100226799136,

0.13297484367902618,

0.05425093840445618,

0.06318158399798929,

0.1820484957396675,

0.020718413413777427,

0.12480061621768092,

0.32878552750252776,

0.053606243238287894,

0.056767905504853014,

0.16355687804587343,

0.12704511790457915,

0.11624486383293112,

0.06676387516971168,

0.05682702546668007,

0.03208146198293162,

0.1801738496423962,

0.12905433400195887,

0.24902334269037305,

-0.0003369652170926024,

0.20820884182144622,

0.02929789085381035,

0.17908579047489231,

0.00029106773147016124,

0.12377626825455652,

-0.0024964545804570318,

0.06494701453114213,

0.13456733474121033,

0.10339406818095989,

0.11740123127732596,

-0.01889294640656,

0.02846073690160248,

0.011861822701336115,

0.16392097125848626,

0.004915696395333256,

0.008225260311491282,

0.043351141729182606,

0.2145841457479266,

0.2196230819339824,

0.10494763638039072,

0.16876850732212983,

0.02288276196547849,

-0.03324720640059757,

0.23711309493003968,

0.22846385079029316,

0.016826019528002438,

0.2232699778788444,

0.14719057196294646,

0.04577324109525298,

0.10044996210059598,

0.12084328845808198,

0.16912127304253724,

0.019570586972487154,

0.07702688824013633,

...]

* 1. Provide the code used to support the implementation of the multiple regression models.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import MinMaxScaler

import statsmodels.api as sm

data = pd.read\_csv('medical\_clean.csv')

pd.set\_option('display.max\_columns', None)

print(data.describe())

def meddata\_preprocessing(data\_num, data\_enc, data\_yn, data\_ord): #This is saved as a function for reusability in later tasks

for i in range(len(data\_num.columns)):

#Compute the mean and standard deviation of each column

mean, std = np.mean(data\_num.iloc[:,i]), np.std(data\_num.iloc[:,i])

#Set the upper and lower bounds at three standard deviations from the mean

upper, lower = mean + 3 \* std, mean - 3 \* std

#Record the index for each row that contains a value outside the previously set boundaries

drop = [inx for inx, x in enumerate(data\_num.iloc[:, i]) if x < lower or x > upper]

#Compare the list of indices to be dropped with those within the dataframe and drop those that remain

#within the data frame while ignoring those that were already dropped

for d in drop:

if d in data\_num.index:

data\_num = data\_num.drop(d)

#One-hot encode categorical variables

data\_enc = pd.get\_dummies(data\_enc, prefix=data\_enc.columns, drop\_first=True)

for col in range(len(data\_yn.columns)):

#Replace values of "Yes" with 1 and values of "No" with 0

for inx, val in enumerate(data\_yn.iloc[:, col]):

if val == 'Yes':

data\_yn.iloc[inx, col] = 1

else:

data\_yn.iloc[inx, col] = 0

#Determine levels of ordinal variables

scale\_mapper = {

"Area" : {

"Rural" : 0,

"Suburban" : 0.5,

"Urban" : 1

},

"Initial\_admin" : {

"Emergency Admission" : 1,

"Observation Admission" : 0.5,

"Elective Admission" : 0

},

"Complication\_risk" : {

"Low" : 0,

"Medium" : 0.5,

"High" : 1

}

}

#Replace values with numerical equivalents specified above

for col in data\_ord.columns:

data\_ord[col] = data\_ord[col].copy().replace(scale\_mapper[col])

mm = MinMaxScaler() #Instantiate the MinMaxScaler method

data\_num[data\_num.columns] = mm.fit\_transform(data\_num) #Standardize the data

#The use of an inner join preserves the dropping of rows performed on data\_num

#The data\_yn dataframe is converted to a numeric datatype, int32, before joining

data\_clean = data\_num.copy().join(data\_enc, how='inner').join(data\_yn.astype('int32'), how='inner').join(data\_ord, how='inner')

return data\_clean #Returns a fully prepared data set

#Call the preparation function

data\_clean = meddata\_preprocessing(data\_num, data\_enc, data\_yn, data\_ord)

data\_clean #Preview the data

#Univariate distributions of all independent variables

fig, ax = plt.subplots(14, 3, sharey=False, figsize=(14, 64))

axi = 0

for i in np.r\_[8:12, 14:50]:

ax.flatten()[axi].hist(data.iloc[:, i])

ax.flatten()[axi].set\_title(data.columns[i])

axi += 1

#Bivariate distributions for independent variable vs. the dependent variable

fig, ax = plt.subplots(8, 3, sharey=False, figsize=(16,32))

inx = 0

for i in np.r\_[8:12, 14:17, 20:24, 39:50]:

ax.flatten()[inx].scatter(data.iloc[:, i], data['Initial\_days'], )

ax.flatten()[inx].set\_title(data.columns[i])

ax.flatten()[inx].set\_ylabel('Initial\_days')

inx += 1

#Bivariate distributions continued

fig, ax = plt.subplots(7, 3, sharey=False, figsize=(14,32))

axi = 0

for i in np.r\_[11, 17, 18, 19, 24, 25:39]:

sns.boxplot(data=data, x=data.columns[i], y='Initial\_days', ax=ax.flatten()[axi])

ax.flatten()[axi].xaxis.set\_ticklabels([])

axi +=1

#Export prepared data to csv

data\_clean.to\_csv('med\_data\_prepared.csv')

#Divide prepared data into predictors (X) and target (y)

X = data\_clean.drop('Initial\_days', axis=1).copy()

y = data\_clean['Initial\_days'].copy()

#Add constant

X = add\_constant(X)

#Instantiate and fit linear model

lin\_reg = sm.OLS(y, X)

model\_init = lin\_reg.fit()

model\_init.summary()

#Check for multicollinearity through heat-map of correlation matrix

plt.figure(figsize=(12, 12))

sns.heatmap(data[data.columns[4:50]].corr(), linewidths=0.5, linecolor='white', cmap=sns.color\_palette("flare", as\_cmap=True), square=True)

#Remove highly correlated independent variables and confirm no strong correlations through second heat-map

X2 = X.drop(['Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8', 'Additional\_charges'], axis=1).copy()

#Model reduction strategy is saved within a function for reusability

def reduce\_model(X, y, sig\_lvl):#Accepts predictor df, target series, and significance level

lin\_reg = sm.OLS(y, X)#Creates model

model = lin\_reg.fit()#Fits model

p\_val = model.pvalues.to\_list()

var = X.columns.to\_list()

while max(p\_val) >= sig\_lvl:#Breaks once all predictor variables are statistically significant

for i in range(len(var)):

if p\_val[i] == max(p\_val):

X.drop(var[i], axis=1, inplace=True)#Drops variable with highest, non-significant p-value

lin\_reg = sm.OLS(y, X)#Recreates model

model = lin\_reg.fit()#Refits model

p\_val = model.pvalues.to\_list()

var = X.columns.to\_list()

return model, X#Returns a reduced model and the corresponding predictor variables

model\_final, X\_final = reduce\_model(X2, y, 0.05)

model\_final.summary()

#Store model residuals for ease of access

resid = pd.DataFrame(columns=['Initial', 'Final'])

resid.Initial = model\_init.resid

resid.Final = model\_final.resid

#Initial model residual plot

predictions\_init = model\_init.predict(X)

sns.scatterplot(x=predictions\_init, y=resid.Initial)

plt.title('Initial Model Residuals')

plt.xlabel('Predicted Output')

plt.ylabel('Residuals')

#Reduced model residual plot

predictions\_final = model\_final.predict(X\_final)

sns.scatterplot(x=predictions\_final, y=resid.Final)

plt.title('Reduced Model Residuals')

plt.xlabel('Predicted Output')

plt.ylabel('Residuals')

#Residual Error of reduced model

np.sqrt(model\_final.ssr / model\_final.df\_resid)

#Removed Variables

print([x for x in X.columns.to\_list() if x not in X\_final.columns.to\_list()])

#Reduced model predictions

model\_final.predict(X\_final).to\_list()

#Reduced model coefficients

model\_final.params

**Part V: Data Summary and Implications**

1. Summarize your findings and assumptions by doing the following:
   1. Discuss the results of your data analysis, including the following elements:

The reduced model's params attribute provides us with the coefficients for our regression equation. We can use these coefficients to generate the following equation:

Where:

, , ,

,, , ,

, , ,

,

We can see from the equation that most of the coefficients are very small, except for the coefficient for TotalCharge. This implies that TotalCharge has the greatest effect on our predicted value, with the other variables providing minimal compensations for the variance in the target variable.

The const variable, representing the constant of the model’s equation, has a small positive coefficient. This is expected as it is likely that hospitals have a minimum length of stay that is non-zero and not negative. This short minimum stay is likely to allow the hospital staff to ensure that danger has dissipated, and the patient will be safe if they leave, or for paperwork to be filed to ensure the hospital is not liable for the patient legally if they opt to leave under their own power.

Readmission also has a positive relationship with length of initial stay, a fact that was discovered in the task for Exploratory Data Analysis. The variable vitD\_supp has a small negative coefficient, implying that patients treated with more vitamin D supplements have shorter initial stays than patients given fewer supplements.

Next, we see a group of diagnostic variables indicating whether a patient has a specific diagnosis. Of these diagnoses, none have a positive coefficient, implying that any diagnosis results in a shorter initial stay. These coefficients imply that when doctors are aware of the underlying health condition of their patients, the root cause of admission is identified more quickly, and the patient cleared for discharge more expeditiously.

Finally, we explore the coefficients of the ordinal variables Initial\_admin and Complication\_risk. These variables were encoded such that higher values of Initial\_admin and Complication\_risk represent higher patient risk levels. The coefficients of Initial\_admin and Complication\_risk are negative, implying that patients deemed to be at higher risk are predicted to have shorter initial stays. This is likely due to the prioritization of high-risk patients leading to faster, more intensive care that leads to identification of the underlying cause of admission which allows for treatment to begin, and thus complete, sooner than patients prioritized as lower risk.

The model's high adjusted R-squared value tells us that the model is statistically significant and provides an accurate prediction of the target variable with the provided data set. However, the practical significance of the model is lacking. Because of the inflated importance of the TotalCharge variable, the model has little real-world application as a patient's balance due, TotalCharge, is calculated based on how long they stayed in the hospital, Initial\_days. One hardly needs to predict the length of a patient's stay once they have completed their visit. Despite this failing, the model helps us to see which other variables are likely contributors to the length of a patient's stay.

* 1. As the coefficients for Initial\_admin and Complication\_risk are the second and third highest coefficients respectively, these variables appear to be the most influential predictor variables that will be available to hospital staff at patient admission. For this reason, hospitals should be aware that, while prioritization of patients experiencing medical emergencies is necessary, lower risk patients should not be ignored. By increasing staffing and resources allocated to non-emergency care, patients who are labeled as lower risk can receive better, faster care. The speed of identification of the cause of admission and its treatment appears to be a major contributing factor to a patient’s length of initial stay. Reducing this may lead to a decrease in the average stay of patients and, as a result, reduce the total number of readmissions at the hospital.

**Part VI: Demonstration**

1. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=425cd376-5f2d-46fb-b393-adad01530fee>
2. No outside sources were used to generate code for this project and all code was written personally by the student.
3. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

John Hopkins School of Public Health (JHSPH). (n.d.). Chapter 10: Variable Selection. Baltimore, MD. https://www.biostat.jhsph.edu/~iruczins/teaching/jf/ch10.pdf