**Part I: Research Question**

1. Describe the purpose of this data analysis by doing the following:
2. Based on the data available in our data set, can we predict whether a patient will be readmitted after being discharged from their initial hospital stay?
3. As there are limits to the number of readmittances any hospital can have without being fined, it would be useful to know how likely any patient is to be readmitted. Through analysis of the coefficients of the logistic regression, we may be able to identify characteristics of patients who are likely to be readmitted and recommend action that allows for hospitals to address common causes of readmittance on the initial stay and reduce the number of readmittances.

**Part II: Method Justification**

1. Describe logistic regression methods by doing the following:
2. Logistic regressions require that certain characteristics of the predictor and target variables be satisfied. The first assumption is that the target variable is binary or ordinal (Statistics Solutions, 2021). Second, samples in the data should not be repeated measures of the same sample; logistic regression is not a model designed for time series data. Third, multi-collinearity should be reduced as much as possible for the model to work properly. Fourth, the predictor variables should have a linear relationship to the log odds of the target variable. Finally, linear regression models are most effective with a large sample size. Specifically, the least frequent category of the target variable must account for at least 10% of the outcomes (Statistics Solutions, 2021).
3. Python, when paired with Jupyter Notebook as the development environment, provides a clean workspace for exploring, transforming, and modeling data. The availability of powerful, open-source libraries expands the capabilities of Python to easily handle many difficult data science tasks. The pandas library allows for structuring of data and easy access of samples and variables within the data set, as well as separation and joining of data into data frames. The numpy library provides useful computation tools, such as the square root function sqrt, as well as tools for creating and manipulating arrays. Matplotlib and seaborn, which is built atop matplotlib, allow Python to create organized and satisfying visualizations of the data. Statsmodels is a library for model creation and evaluation that provides detailed information concerning the effectiveness of the model, the coefficients, and other statistical measures and scores of the model. Finally, sklearn, better known as Sci-kit Learn, provides model development tools as well as evaluation metrics. For this project, we use sklearn for its extensive catalogue of data preprocessing functions.
4. Logistic regression is the ideal regression for analysis of binary target variables. The target variable for our analysis is ReAdmis, a binary variable. Our data also satisfies, or can be coerced to satisfy, the assumptions of logistic regression. The data is not time series data and variables that are multi-collinear or do not have a linear relationship to the log odds of the target variable can be removed to ensure the model fits well.

**Part III: Data Preparation**

1. Summarize the data preparation process for multiple regression analysis by doing the following:
2. Summarize the data preparation process for multiple regression analysis by doing the following:

The data preparation steps for this task are completed through the application of the preparation function created in Task 1, with the same objectives and processes.

1. 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 through the use of 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 of 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 still 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 preperation 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:

Code:

#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

Output:

Chart, histogram

Description automatically generated

Chart, waterfall chart

Description automatically generatedGraphical user interface

Description automatically generatedChart, waterfall chart

Description automatically generatedChart, histogram

Description automatically generated

Bivariate:

Input:

#Bivariate distributions for independent variable vs. the dependent variable

fig, ax = plt.subplots(7, 3, sharey=False, figsize=(24,52))

axi = 0

for i in np.r\_[8:11, 14:17, 20:24, 39:50]:

sns.boxplot(x=data['ReAdmis'], y=data.iloc[:, i], ax=ax.flatten()[axi])

ax.flatten()[axi].set\_title(data.columns[i] + ' by ReAdmis')

ax.flatten()[axi].set\_xlabel('')

ax.flatten()[axi].set\_ylabel('')

axi += 1

Output:

Diagram

Description automatically generated

Diagram

Description automatically generated

Input:

#Bivariate distributions continued

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

axi = 0

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

x = data.columns[i]

y = 'proportion'

hue = 'ReAdmis'

hue\_order = [0, 1]

(data[x]

.groupby(data[hue])

.value\_counts(normalize=True)

.rename(y)

.reset\_index()

.pipe((sns.barplot, 'data'), x=x, y=y, hue=hue, ax=ax.flatten()[axi]))

ax.flatten()[axi].set\_title(data.columns[i] + ' by ReAdmis')

ax.flatten()[axi].set\_xlabel('')

ax.flatten()[axi].set\_ylabel('')

axi += 1

Output:

Chart, bar chart, treemap chart

Description automatically generatedChart, bar 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 logistic 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 Logit method is then used to generate a logistic regression model which we fit to the target and predictor variables. The summary method of this model provides information on the goodness-of-fit and each variable’s statistical significance to the model, among other metrics.

Input:

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

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

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

#Add constant

X = add\_constant(X)

#Instantiate and fit logistic model

log\_reg\_init = sm.Logit(y, X).fit()

log\_reg\_init.summary()

Output:

Table

Description automatically generated with medium confidence

Table

Description automatically generated with medium confidenceTable

Description automatically generated

The output shows that the logistic regression fits decently to the target variable with a pseudo R-squared of 0.9506. However, this initial model has not been verified to satisfy the assumptions of a logistic regression, and the column of p-values displays many statistically insignificant variables.

1. Two methods of feature selection were employed 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 by 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 Item variables all have correlations with one another,. In order to remove this multicollinearity, the Item variables, and TotalCharge were 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, it will be dropped 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

log\_reg = sm.Logit(y, X)#Creates model

model = log\_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

log\_reg = sm.Logit(y, X)#Recreates model

model = log\_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(Xc2, y, 0.05)

model\_final.summary()

Output:

Graphical user interface

Description automatically generated with low confidence

We can see in the output that all of the variables have p-values less than 0.05 and can be deemed statistically significant. The pseudo R-squared value has dropped slightly to 0.9471, indicating that the final model fits the target variable slightly less accurately. The most significant variables, according to their coefficients, are Initial\_days, Services\_MRI, and Initial\_admin..

1. Analyze the data set using your reduced logistic regression model by doing the following:
2. Explain your data analysis process by comparing the initial and reduced logistic regression models, including the following elements:
   1. Feature selection is accomplished 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 10 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', 'vitD\_supp', 'TotalCharge', 'Additional\_charges', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8', 'Marital\_Married', 'Marital\_Never Married', 'Marital\_Separated', 'Marital\_Widowed', 'Gender\_Male', 'Gender\_Nonbinary', 'Services\_Intravenous', 'Soft\_drink', 'Overweight', 'Diabetes', 'Hyperlipidemia', 'BackPain', 'Allergic\_rhinitis', 'Reflux\_esophagitis', 'Area']

In order to compare the initial and reduced models, we will look at the pseudo R-squared metrics of both the initial and reduced model. In the initial model, this metric is 0.9506; in the reduced model, it is 0.9471. What we see is a slight decrease in this metric, which is used to evaluate the goodness-of-fit of the model, implying that the reduced model fits the data very well, though not quite as well as the initial model. This could be an indicator that the model is over-fitting the training data as a Pseudo R-squared of 0.95 is almost too good.

1. Reduced Model Coefficients:

const -76.757150

Initial\_days 99.085875

Services\_CT Scan 1.761314

Services\_MRI 2.474357

HighBlood 0.790651

Stroke 1.529204

Arthritis -1.237706

Anxiety -0.867477

Asthma -1.282136

Initial\_admin 2.442183

Complication\_risk 1.841626

Initial Model Confusion Matrix:

model\_init.pred\_table()

array([[5682., 62.],

[ 61., 3318.]])

Reduced Model Confusion Matrix:

log\_reg\_final.pred\_table()

array([[5678., 66.],

[ 58., 3321.]])

Reduced model predictions:

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

[6.425822161090203e-28,

2.7143904697223423e-23,

4.940986616185922e-31,

1.1659355242947253e-33,

3.828266109318979e-33,

1.1526345407901607e-30,

1.0933071465035628e-28,

4.106813423365349e-25,

7.326241514115066e-30,

4.94394086045855e-32,

7.554752471601485e-32,

8.297449041166499e-28,

1.0416203321917294e-30,

5.831274724816067e-32,

1.6499990639864925e-33,

3.444127434703866e-33,

1.6175625513722707e-32,

4.900823626819943e-30,

6.404619075115286e-29,

3.6499801976840405e-28,

1.0636136572159984e-30,

8.329835752660624e-26,

1.527634271562488e-22,

2.573968554968681e-28,

4.860553564181159e-29,

7.082431782266129e-22,

5.287456422682586e-26,

1.0125486969480845e-25,

5.494039438554215e-31,

4.4262174950459954e-18,

2.8450701366406856e-30,

8.69244439941904e-28,

5.383287931396745e-27,

2.3044797451718728e-32,

1.6792459803879815e-26,

4.928705129739464e-25,

3.879933264894059e-27,

3.393114941138485e-32,

3.367817497408879e-28,

1.053748458345059e-27,

1.225994599133899e-25,

1.7357145558703548e-23,

3.366901521814933e-27,

3.9411897025519617e-26,

3.478785710205938e-32,

6.185783063368789e-25,

7.702530127479266e-24,

4.7294585255757346e-24,

1.671709706335288e-26,

5.389727725075406e-32,

2.3180667271203283e-32,

1.196614782536939e-31,

7.313455555489826e-31,

6.746726941648264e-27,

1.676565054758489e-26,

2.7660334891567933e-18,

7.057399134969368e-30,

1.2817243242122636e-19,

3.724433222536995e-33,

1.394490506204853e-29,

2.2623022946262723e-14,

3.1492238224616256e-27,

9.53023036851088e-28,

1.1173838414844225e-31,

1.050012208457088e-28,

4.0083378537115087e-22,

1.2424989759877045e-21,

1.2868537947097903e-31,

5.83649896095188e-30,

1.746795026191495e-30,

3.4410005105615846e-27,

5.329021437504749e-28,

5.132058757010952e-28,

1.7902247779601174e-28,

2.4350934399995436e-28,

4.001881321886512e-31,

2.417888390804175e-21,

1.472805544475144e-31,

1.604206237521488e-26,

7.625286268886333e-28,

1.3397128814732298e-31,

1.3267530984782069e-32,

1.5406960339901902e-26,

2.2723434900081516e-29,

1.1147199614839806e-29,

1.5765647601807096e-26,

2.606410825283885e-32,

4.080339278940738e-34,

2.4965568062130244e-32,

1.4290637898332356e-28,

1.2083486677778516e-29,

3.7275427120894847e-31,

5.274594802490585e-28,

1.2086463865794022e-18,

5.879157824354374e-35,

5.365258543026144e-28,

1.413887680723143e-19,

2.6659437758209917e-28,

4.1784049473312687e-32,

8.281865033001732e-28,

1.3367317798525293e-30,

1.0208810671705297e-29,

1.86950044032341e-30,

1.5273313713321946e-31,

1.5119952683397939e-30,

3.0296657162462805e-31,

3.380305823407999e-29,

1.2891979716406586e-28,

1.0349411095023856e-32,

9.091650529775846e-30,

1.7993265177761098e-25,

4.204352652811185e-30,

1.3318820998933254e-26,

1.3627903260532513e-26,

1.8733468621260808e-28,

4.0761963100748496e-26,

1.1733906010338767e-31,

1.1564807061846174e-32,

4.4337221862022035e-27,

1.4716582160587595e-33,

3.903923583373132e-31,

1.5933876886897644e-30,

6.089009555493777e-29,

8.286986601066547e-27,

7.578786885929715e-31,

4.012091213451047e-30,

1.3818652305711659e-28,

1.754757306257744e-32,

3.42958063594757e-32,

8.305932862313197e-33,

7.781585600194333e-31,

6.761766720627435e-25,

1.8776097402991676e-27,

1.4596056346628145e-30,

1.868069888190396e-27,

3.6226180887021848e-28,

2.9524111904247852e-30,

2.8527934527655426e-31,

1.7727534144396581e-22,

1.0963478002491262e-15,

5.898256947246605e-26,

8.412061625679878e-22,

1.0313290885979919e-31,

1.351704398788686e-31,

6.887826311809842e-31,

5.078138665505756e-25,

1.4990608872962013e-24,

1.1816227347793613e-29,

6.393849103797485e-27,

2.345876728745052e-24,

5.0643287383514814e-27,

4.292693860431651e-24,

1.3855919362416258e-22,

4.593707501241062e-25,

2.6876107382816013e-31,

2.6712651407130027e-31,

1.59871797085828e-33,

1.0349066438183194e-23,

1.0696907590544373e-19,

3.150192371583201e-33,

1.5660971884654862e-31,

2.1759360138289396e-20,

8.556395253415996e-33,

2.720153979744755e-29,

3.467077930722803e-27,

1.5643766820647845e-23,

5.0317473770677927e-26,

2.405715381206354e-28,

9.991555974445274e-32,

1.5734607688870628e-29,

1.0315823656722111e-33,

6.0203321844312935e-30,

1.9000596003778783e-31,

7.958916665748272e-31,

1.2761010747716818e-29,

1.0084270471078901e-28,

1.5384445120816169e-18,

4.9610383904311515e-31,

1.9722442958058266e-26,

8.468450277304144e-33,

3.015915463471369e-28,

1.3881946833337844e-26,

8.332994947228054e-33,

2.591611451362441e-33,

1.8584830915056302e-29,

4.229025562863319e-26,

5.651965551115339e-26,

1.9228918866159065e-26,

2.2240874216130205e-28,

1.186624369075925e-25,

1.0979895851981276e-21,

2.4162243971821206e-32,

3.8712313683354987e-33,

9.089934985595623e-26,

1.0156358411416344e-30,

1.2076938685925475e-26,

6.070049103715603e-27,

2.8719033513330256e-29,

1.0748682928656562e-20,

2.1562778122898932e-30,

1.2451366143185317e-27,

1.3929443063380375e-25,

6.701780790255655e-27,

6.942261222310747e-24,

4.570241179664268e-30,

9.727312105512839e-32,

3.263319284857037e-27,

2.9461865852484426e-30,

6.573255147581273e-27,

4.728763816262831e-32,

3.0389258831323124e-29,

1.9915496088727558e-30,

6.640014282598251e-34,

1.533947114101928e-26,

4.664289173715605e-25,

1.135020247375895e-29,

1.0781778234453241e-28,

1.1758754923132951e-32,

1.0079210365081663e-28,

3.2537027124547873e-29,

1.0623383206939219e-22,

2.4325222986779126e-30,

5.430652078367516e-26,

6.874923316273451e-27,

2.003864158075729e-30,

5.261189810339262e-23,

9.532145093867986e-28,

4.285019771147236e-28,

8.05075125766966e-31,

2.61928374973952e-26,

6.4410963340079695e-31,

2.7335401044579246e-27,

9.519513020759467e-19,

8.574008209718074e-27,

3.045925848767973e-28,

2.403637479501841e-28,

1.7615811181203745e-32,

1.5021720726240323e-25,

3.0544512193072756e-24,

1.1993207068166041e-24,

6.644343835786787e-27,

4.973386276218539e-23,

2.2661305820931987e-29,

1.0538230844631706e-19,

6.238567435082093e-32,

7.632117966973437e-29,

1.439680824899062e-24,

2.274036927959509e-30,

1.1995965138040278e-29,

6.443898183232858e-32,

2.194850621250779e-30,

1.110251167896668e-20,

8.026371750485718e-29,

5.4903268176320825e-28,

5.8168535650483176e-33,

1.6892334428121528e-26,

1.139593928179889e-29,

4.710787528653501e-32,

2.4816729319766935e-32,

4.646074035434526e-23,

9.978500274242681e-29,

1.7834776892890726e-17,

3.069104010751985e-29,

4.009595580377925e-30,

1.9635014675035555e-25,

1.5921109335583228e-29,

4.869581464967512e-30,

3.202985140318557e-29,

2.885362207818307e-26,

1.580370356845053e-25,

3.663833068690309e-19,

4.0891157561414266e-29,

7.268596682876369e-21,

1.9204523715533064e-31,

1.3095543501304815e-31,

1.6744129557699675e-30,

9.267934367333842e-28,

6.094676663695523e-28,

4.337489998772241e-27,

1.7941647043958736e-25,

4.460687325297194e-29,

8.063135113794452e-27,

5.980374135892228e-31,

1.0411163921391762e-22,

9.599927709964338e-32,

2.6317695027109493e-19,

1.8496271346561741e-31,

1.1034318745044727e-26,

7.967562472312441e-28,

5.641715674677127e-33,

4.338070704896089e-19,

1.0563218769873436e-26,

3.844050558995547e-31,

1.2742227270381476e-25,

3.956734207733623e-17,

4.492299619892591e-24,

1.5737472441811694e-16,

6.520651537230458e-28,

1.3322037419396243e-29,

4.511498808101886e-23,

5.755965119103033e-31,

1.8283551021052802e-20,

1.779050822464421e-32,

7.928046489142006e-24,

2.680120220189741e-28,

3.5251866772277417e-26,

1.1724423370912545e-21,

3.7858359816867354e-31,

1.9622908448190134e-31,

9.722412525462842e-31,

6.467625785501321e-33,

1.7393694699197914e-27,

3.5192836143671225e-27,

1.381352392738355e-28,

9.478758612433738e-34,

1.5367149692282634e-20,

3.653602065933738e-31,

1.4140215081930948e-28,

3.8303437940161703e-25,

1.2685421420234762e-31,

3.548425247325486e-33,

3.779492799460223e-29,

1.4237439550852798e-27,

2.9306228908946496e-32,

3.049625483492772e-28,

3.2813883597169814e-30,

5.077748902141259e-25,

9.808220902582795e-28,

6.769783783962692e-29,

2.0136951565251075e-24,

7.561950547878665e-28,

2.3200211555160533e-29,

1.775562993307738e-29,

1.1388385694118294e-26,

1.9767342783583885e-31,

5.2012916534689254e-27,

1.43268423444036e-32,

8.693155005251139e-32,

2.9984157722952966e-29,

1.5537334584663049e-31,

3.9045337824736496e-32,

1.786066020316984e-23,

1.2064157474346909e-28,

1.0124819965982361e-26,

2.1983724224153984e-29,

5.487938835107307e-29,

6.304487155400728e-27,

4.27986510647612e-30,

7.635034634523036e-32,

5.498534288048318e-31,

3.341875218279435e-29,

1.5378526379086365e-26,

4.409368551206603e-25,

1.8517111904068927e-19,

2.0886373785007696e-30,

6.682340290632676e-33,

7.001002342422669e-28,

5.638964344988881e-31,

1.6151510752204827e-27,

3.0393887457913846e-22,

4.0526917718308016e-26,

3.004156110680715e-29,

8.754389113353861e-31,

4.107327076298554e-24,

9.372048799222389e-33,

1.566329195809727e-31,

6.148130158080592e-28,

1.3452718058629907e-31,

5.00903748417738e-28,

5.000634932136088e-22,

2.923410363660613e-25,

4.136641677974457e-22,

1.869951770668169e-24,

6.3656135468101205e-31,

7.427636201771014e-31,

5.211066224805061e-24,

2.54865540471804e-33,

1.8477202536116885e-31,

9.316865839105261e-33,

2.975149358080028e-29,

1.5866704075162663e-25,

5.902335852579731e-21,

6.318110285273898e-28,

4.685627763515198e-32,

1.6945570177795607e-32,

1.6414902583459199e-24,

1.2825677524307573e-31,

4.006536890815043e-27,

1.272105401784495e-33,

1.2631580189821795e-29,

2.652375005152811e-27,

3.893654694317748e-24,

7.139545473866491e-30,

1.3870014938566547e-32,

3.4751861902207397e-22,

1.744974597449369e-29,

2.507505922265976e-27,

8.62224662491944e-27,

6.313174006973658e-19,

2.0532551447441322e-29,

2.707957725900812e-32,

2.6894569227825165e-28,

2.2556526957113488e-29,

2.0423260049187808e-28,

3.025663169560387e-28,

4.454222958290952e-33,

7.12119530468789e-26,

1.1707216480312794e-29,

5.503078215449192e-24,

5.28063332628868e-30,

3.402118610357915e-27,

7.642674426444279e-26,

5.354447437201324e-28,

1.2400646523176876e-25,

4.452279085686824e-22,

5.170589684196534e-30,

3.760667118433468e-18,

4.635372396663856e-29,

1.0821828677268195e-21,

7.322317166504232e-29,

1.5889441570829824e-24,

3.9048123578135053e-28,

2.1298055798002866e-31,

4.306166036238158e-32,

2.9902461141177545e-31,

8.227757500997431e-30,

3.810928779236613e-27,

8.856536441245064e-28,

9.222474707080495e-33,

2.315808459571167e-23,

2.9279695988778014e-19,

5.176915922346612e-32,

1.0383702397157096e-31,

1.4360449626687852e-28,

9.76008487400887e-34,

4.8480111600936027e-32,

1.8187237628929307e-31,

2.921598015274521e-32,

6.880750176348215e-29,

4.812200804302917e-24,

4.4429056910218464e-23,

4.1408146713450823e-32,

8.91227168274614e-31,

1.5619318444748522e-22,

4.2692253032836724e-27,

1.374573933936591e-25,

2.1131191784906166e-30,

2.4674336982141706e-31,

3.991586800197284e-29,

1.2324848763445381e-26,

1.7965192269917698e-21,

2.0895856636599194e-31,

5.387293144343049e-27,

1.577933382334595e-16,

9.556328888333685e-30,

3.5944523937926673e-19,

6.891900517349247e-33,

4.872851409284728e-32,

5.233275454557424e-25,

3.2300628417702947e-24,

1.9170483796306988e-26,

1.7339965703705868e-23,

5.7031891501937795e-27,

1.2249603893663247e-27,

1.5467876520520895e-19,

8.896765414730676e-28,

8.32066021511927e-30,

2.044116665228556e-28,

2.410549413579318e-29,

9.420265173975351e-32,

2.1451878142328243e-26,

1.1564143305483347e-22,

1.0110028031993336e-32,

5.588076790117813e-21,

1.684300440565886e-23,

1.2341808238006953e-25,

1.0335225064862397e-27,

1.0458370438708047e-32,

4.613009217954232e-32,

1.9981780911779341e-28,

2.6755627884486775e-32,

6.931740530303215e-31,

1.5816943266955378e-30,

9.782293184635888e-32,

3.0958645415418884e-23,

1.1768123481902143e-26,

8.565840429481364e-29,

2.125114959264627e-28,

5.898003298998052e-25,

9.461880434397161e-30,

7.937965432853835e-30,

6.801245474277389e-32,

1.6049859986386686e-33,

1.6647697649423994e-21,

1.6494661024256308e-31,

2.8649897689265305e-33,

3.0815456520894216e-21,

9.965251459252579e-26,

1.438951292514892e-26,

5.6559009715828145e-31,

5.090469275666874e-34,

1.6333200121341853e-23,

1.6392558394193492e-14,

1.4902348114618958e-21,

1.4378937588304432e-31,

9.790805611910929e-28,

9.625775732448015e-27,

1.4302223917472693e-28,

5.541416375674628e-20,

3.8159207108718037e-29,

4.0402571936595225e-30,

4.14074206693219e-27,

1.7358653171513746e-33,

2.2319269922496582e-30,

3.0323932397878136e-24,

2.0104233708062267e-24,

3.0621301328177213e-29,

1.1405365466551213e-28,

1.1473664649127597e-25,

6.249055498696186e-23,

9.454410225021902e-27,

4.3407969579339687e-29,

6.856539573866557e-23,

1.2437788873498927e-28,

1.0024009668807135e-23,

1.870944628798938e-27,

3.537952378212644e-25,

1.028509479606775e-20,

5.1143515519465355e-33,

3.992918120509819e-25,

3.764625672066175e-30,

8.491742030783246e-24,

7.161725767952431e-33,

4.945312140225931e-27,

1.458216218985434e-25,

1.5856597468099403e-23,

9.703251228964564e-29,

1.798976759817583e-30,

1.2630682731821522e-23,

5.120964518918729e-32,

3.952627100069886e-31,

2.0245746064424185e-28,

4.223116434659039e-28,

1.061655977118388e-29,

6.147857342248763e-27,

5.58994199126562e-29,

3.407710752206566e-29,

3.0126278897091424e-30,

4.088848608099627e-26,

1.331178878769883e-23,

2.037601972087384e-25,

1.6220937954600808e-30,

8.149390220967349e-27,

5.002448224049375e-17,

2.3739711925389374e-29,

7.022224092558766e-33,

3.1318329334865164e-30,

2.587528386915731e-26,

8.181180894597597e-32,

3.261108470072412e-27,

7.691901706931411e-26,

1.4133112342984554e-26,

1.9087053768706842e-29,

1.9771982078658354e-31,

2.2015670260323937e-27,

3.0149687266119532e-25,

2.494264689043181e-31,

2.433308571226996e-27,

7.952126457915949e-26,

1.0579346833759266e-29,

1.7743530842854095e-27,

7.270626747637125e-33,

3.129647914609531e-29,

6.970624251444564e-25,

1.367500372554541e-27,

2.1327780270511203e-25,

8.971232341838702e-29,

3.6195854905128154e-25,

1.7343881913840798e-23,

4.033478843529567e-31,

5.318092026247278e-25,

1.0211805952858336e-31,

1.0552697804407726e-29,

7.680845317393554e-24,

2.351388144646342e-30,

3.242271987126862e-27,

1.3199280439158123e-31,

8.93678114633571e-31,

1.7859648727842355e-28,

4.7907705378926045e-32,

2.6166235298485253e-33,

5.347284485268283e-29,

8.40183246215904e-25,

8.396301578302927e-30,

4.178327988315639e-23,

8.015439059901714e-31,

5.449257152197295e-29,

2.760418435953758e-29,

7.236026850311832e-30,

3.665624352718432e-28,

5.7400604309332586e-24,

4.847027563156072e-14,

2.114858323628523e-23,

2.226896681779458e-22,

1.5866166605618568e-25,

2.970678019446517e-31,

4.421053693403223e-32,

5.491638977784083e-28,

2.5285290575731163e-32,

1.5386508819818317e-24,

1.5575098555522378e-24,

2.2817191784361602e-30,

1.2829251831223536e-19,

5.619402901676649e-31,

1.8932702030633154e-24,

5.04987113294906e-30,

1.931983481804479e-28,

6.043414157074003e-33,

1.9003090866310485e-24,

1.0443829249747307e-23,

2.0995590190340447e-31,

9.994959823072407e-32,

1.4412551469837512e-26,

8.093607569198986e-26,

5.224155765734885e-29,

7.16604641176213e-28,

1.3745816662987034e-26,

6.606540434928399e-32,

3.7630171392074777e-28,

9.440606524249975e-27,

2.178817370387064e-29,

2.6159877403134903e-33,

5.221091650153462e-32,

8.368255520130534e-26,

1.3388711910957327e-32,

1.1525280675790747e-32,

1.3449785437719534e-29,

9.377476496688089e-32,

4.090584646503495e-28,

1.0893094958276042e-18,

1.4468352306413864e-29,

6.487400027807294e-21,

9.533980359676094e-19,

7.9283021525829e-31,

1.613802469522753e-26,

9.143243224159926e-32,

8.00262764065888e-32,

1.6701625079627484e-28,

6.448989216753173e-22,

3.3744116423950294e-31,

2.4685510236676152e-29,

1.3377069291696422e-26,

3.1186015893629913e-32,

9.132693715363566e-32,

1.0185600411258393e-26,

7.369511378247097e-15,

2.2213347577428863e-23,

1.5515577114276904e-31,

2.9207105796251465e-28,

4.635814612536579e-26,

3.1445993038109973e-19,

3.411572346567909e-30,

8.975127657720262e-28,

1.9066732267680328e-25,

4.9032002738269386e-29,

1.514064017171404e-30,

4.2035232878232324e-30,

7.640732201594644e-24,

3.1000175578426976e-23,

1.0812025549030889e-23,

2.2216651232736157e-31,

9.294577852807098e-27,

4.373388593579436e-31,

3.9146608934320313e-31,

1.0858076433647222e-28,

1.234359011997438e-28,

4.609829597434711e-32,

3.01693726709137e-31,

2.150146628114005e-27,

2.1474648003662205e-23,

2.4302477564413567e-21,

6.445181881754295e-23,

4.7760037419221494e-29,

2.7496707709994733e-28,

9.584630637493516e-28,

2.1712230713863223e-28,

1.0350882418124052e-28,

4.498849560556784e-24,

2.5058255535107086e-29,

2.1505306385099435e-29,

7.806125118197767e-29,

1.5655596825527691e-30,

1.616374992032414e-30,

3.534949950461802e-24,

6.403558935290673e-21,

9.3083716963837e-34,

1.7003968230646848e-31,

1.792344035510122e-30,

2.2434187005668767e-30,

8.228046663906267e-34,

1.903730469196573e-27,

7.304054367963689e-31,

6.493160267375311e-33,

9.903595950294739e-27,

5.305881785304258e-31,

6.838431684938236e-25,

1.456478944228535e-25,

1.0332396278067223e-25,

8.517798643082918e-27,

2.17051310977787e-26,

1.9535468073495033e-27,

2.0197459869091176e-28,

3.951184921027045e-26,

9.554591872491735e-23,

3.0156886925860087e-32,

4.261931758931945e-20,

4.459008133702086e-26,

4.881839619557624e-27,

1.360537735323652e-30,

1.1663280276410114e-26,

2.4380797826110482e-32,

3.5274131755633343e-31,

2.0033227013734165e-31,

2.0373392938473164e-31,

4.136916518406961e-32,

5.4505565761686745e-33,

1.0113855615069556e-29,

2.636093438446178e-32,

1.0954749345235071e-28,

2.42761316880682e-28,

1.4211381662807364e-30,

6.012466239022374e-28,

3.0441633984376524e-21,

1.900694879755055e-29,

3.6524452964917075e-26,

9.624339795627199e-25,

2.992165778091794e-28,

3.6400637571855334e-31,

3.3260303741842125e-25,

2.0848806120385293e-32,

3.8527390507728743e-22,

3.1308631752359706e-23,

7.58592842806269e-26,

2.045758715996927e-26,

5.520509626953373e-29,

1.8553868102448418e-24,

9.73609007660224e-31,

4.7842262811171567e-23,

2.2577864827725246e-32,

6.283640863980319e-29,

2.748173381356829e-32,

1.7628874956416914e-28,

4.072975479441148e-31,

6.249192620287488e-30,

4.6899408587935286e-24,

1.4508409669742308e-33,

9.105530425724648e-22,

3.466882842126221e-29,

1.2975675091595476e-31,

3.949957665377634e-31,

5.141328986603684e-28,

6.322606226764e-29,

5.078090673506811e-26,

6.618811079378138e-25,

4.936255768033921e-32,

2.5597304031785955e-27,

1.1983209170521964e-30,

2.819903263178258e-26,

2.5034933365901774e-24,

8.082535117862347e-30,

4.188472120418396e-26,

1.4202398493048276e-26,

1.9804660968629606e-27,

2.1199395251672733e-19,

7.38356843583174e-30,

1.3594259061158764e-31,

1.115620055389894e-33,

4.595747371366449e-28,

4.184860315024465e-23,

1.0467055553189778e-21,

7.205315371144189e-33,

9.516599306356336e-32,

4.009109314531729e-32,

3.127204571905123e-24,

5.409047301315359e-30,

5.8562510424360414e-31,

7.7947563473312126e-31,

1.7918261372880155e-27,

4.333821015339978e-30,

2.1969974733124305e-23,

1.5243056236469264e-30,

1.2119892631772641e-29,

7.122868942914728e-32,

1.6088969266789835e-31,

1.2399564955553058e-25,

4.016355427134135e-17,

1.3761662183117975e-29,

3.384515232321542e-27,

4.2209658352546354e-29,

2.7709109252985325e-31,

5.1081567488913367e-26,

1.379723359255807e-26,

6.318380711626157e-28,

1.556558374980266e-32,

1.270000566269478e-29,

2.8926930977064563e-28,

7.264483159345582e-29,

1.5109495289917537e-25,

1.2867502182672867e-28,

8.29799172301979e-28,

8.118822050588741e-27,

1.9570236778233462e-26,

9.713662638579957e-28,

1.8742942752783348e-31,

3.837106336868578e-25,

1.2207648923320247e-30,

4.334611897007583e-30,

9.860580896707073e-32,

8.95697043188765e-25,

2.1527132016778588e-29,

1.705517021634467e-29,

4.869220099881891e-24,

7.142161286424759e-31,

8.54091709980675e-32,

6.408679620784927e-23,

3.271298980752382e-30,

1.233548330980265e-30,

8.46295700685757e-31,

1.6234998983825693e-29,

2.7804346794868617e-26,

4.706320332563237e-25,

3.5319448650621684e-22,

1.1417513588731188e-31,

2.343629619014739e-32,

9.299466134829212e-33,

3.8180902232273174e-30,

6.815472376725734e-29,

7.650269864607354e-31,

9.938566383462766e-21,

2.084197044543163e-29,

1.495288625488204e-26,

1.7858513451693098e-23,

4.833355779846722e-29,

9.119730027604145e-28,

3.170228006227627e-32,

4.259707551714988e-29,

5.694878111022661e-28,

1.314952709469939e-29,

3.1979611600333824e-29,

1.1730726024528335e-26,

1.5336122502122652e-33,

7.785241223356034e-30,

4.627931740446318e-33,

1.1303850494510373e-29,

4.0976363612234115e-27,

4.171356783255181e-29,

4.098482046158209e-27,

1.6493077272397135e-27,

1.620182519929032e-28,

4.6795810915868004e-30,

3.749121685555611e-27,

3.485604066937192e-32,

2.5874403045728693e-31,

8.099448278158343e-33,

4.55682274396264e-25,

7.918104276430363e-31,

3.555379421251597e-26,

2.304615800965957e-30,

1.0376923349697212e-28,

2.642036042521939e-30,

5.491684322517337e-27,

1.7549690163646926e-30,

1.1240713456176104e-29,

1.364916438116717e-34,

1.2700661449249327e-33,

1.238476864468304e-29,

5.975977970907308e-26,

1.5541165110420198e-29,

1.5379759121682578e-28,

2.042099494409489e-24,

2.0761083377673243e-28,

1.8890534627856407e-24,

5.752960705644715e-32,

1.7312858054673872e-33,

4.670140524711186e-13,

1.0841723165436702e-25,

4.543169693837791e-23,

9.335805046986921e-30,

3.6877514652386395e-13,

1.5644418967468728e-31,

3.247171587110293e-25,

2.554406059182333e-31,

3.052719725542057e-27,

1.0774844050316469e-14,

1.5375616802515722e-29,

8.061551785792664e-31,

1.5310603982063757e-26,

2.8701916210674177e-31,

2.3614369686182896e-31,

1.4085060360219994e-28,

1.7712887874895547e-21,

1.4343232570304312e-31,

4.2102880329159036e-24,

1.7875350713895718e-27,

8.589191725520412e-22,

2.667643206769489e-32,

8.71932961099984e-34,

1.5503883884951125e-28,

1.0223647397423709e-30,

4.43431888510851e-31,

1.0271720417234222e-30,

5.564550157614775e-30,

2.340379612667926e-31,

3.7080557507943394e-32,

1.0735231432809033e-30,

1.39430519737332e-30,

1.803611346336021e-22,

5.5542610481382295e-27,

3.7702835313136874e-30,

1.7199819928073466e-26,

1.5810726523181533e-31,

2.8177654962144616e-29,

1.0659853784796689e-30,

8.885187738707478e-28,

1.6036095981178644e-23,

2.2704154462052823e-29,

1.0873082843202527e-28,

4.933361090512525e-32,

8.604789154993044e-30,

1.636545828154826e-32,

1.588745439849019e-26,

7.138688521074565e-32,

3.8961140945338846e-29,

5.730009977570295e-20,

1.5777194532167218e-20,

3.6167758527248835e-25,

2.8321095790327952e-27,

1.9151556402461973e-26,

3.168823457481171e-23,

9.707429362702232e-33,

5.104966202966012e-16,

2.5631982373436053e-29,

6.38242774348092e-31,

1.5869832137763495e-24,

1.1133568383494016e-29,

2.1152574844613392e-29,

3.681896694089572e-30,

2.3663517951541515e-27,

1.660763461063349e-30,

4.0110692045441226e-29,

7.263656535993416e-24,

1.4095506115502372e-33,

4.2441855759902325e-27,

3.3776292387731946e-21,

1.6451456542758057e-30,

1.594048079013094e-29,

3.3939817666709177e-29,

8.855416466043086e-28,

4.3954602014771295e-28,

1.0093381705822063e-31,

1.7042751326995258e-32,

1.3012759260620264e-31,

5.566233914463649e-27,

1.779373699105075e-25,

8.704001698735133e-21,

8.524634637609722e-32,

8.506735116489999e-26,

8.081125961828913e-33,

1.635288616315365e-26,

1.550900788291612e-32,

1.2562467454996903e-28,

6.485336446694625e-33,

8.281192351494959e-29,

5.626933059504932e-31,

3.812953693710057e-28,

2.0294543877132684e-27,

1.3192681850948887e-31,

4.771300931520133e-30,

2.3621729546782835e-32,

1.619107674315251e-25,

5.738888418545333e-32,

3.8041765666365736e-33,

1.1017465903341029e-29,

2.3138339435681284e-26,

1.2574834611823393e-23,

1.8707975537798477e-30,

1.595323945121947e-27,

1.4463423830120124e-31,

9.10491028378312e-33,

6.153128934575343e-23,

8.311933247073342e-22,

1.6101280612714034e-31,

3.521769716632306e-23,

2.2685700852729548e-24,

8.534464316223926e-33,

3.2310541203903104e-29,

2.609032830660875e-27,

1.1110250221471079e-23,

1.8101344073925257e-32,

2.032964161429836e-30,

...]

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

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

from statsmodels.tools.tools import add\_constant

from sklearn.preprocessing import MinMaxScaler

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

#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

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 still 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

#Process and preview the data

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

data\_clean

#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(7, 3, sharey=False, figsize=(24,52))

axi = 0

for i in np.r\_[8:11, 14:17, 20:24, 39:50]:

sns.boxplot(x=data['ReAdmis'], y=data.iloc[:, i], ax=ax.flatten()[axi])

ax.flatten()[axi].set\_title(data.columns[i] + ' by ReAdmis')

ax.flatten()[axi].set\_xlabel('')

ax.flatten()[axi].set\_ylabel('')

axi += 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, 24, 24:39]:

x = data.columns[i]

y = 'proportion'

hue = 'ReAdmis'

hue\_order = [0, 1]

(data[x]

.groupby(data[hue])

.value\_counts(normalize=True)

.rename(y)

.reset\_index()

.pipe((sns.barplot, 'data'), x=x, y=y, hue=hue, ax=ax.flatten()[axi]))

ax.flatten()[axi].set\_title(data.columns[i] + ' by ReAdmis')

ax.flatten()[axi].set\_xlabel('')

ax.flatten()[axi].set\_ylabel('')

axi += 1

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

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

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

#Add constant

X = add\_constant(X)

#Instantiate and fit logistic model

log\_reg\_init = sm.Logit(y, X).fit()

log\_reg\_init.summary()

#Heat map of correlation matrix

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

sns.heatmap(data.iloc[:, 4:50].corr(), linewidths=1, linecolor='white', cmap=sns.color\_palette("flare", as\_cmap=True), square=True)

#Drop highly correlated variables

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

#Redraw heat map of correlation matrix

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

sns.heatmap(X2.corr(), linewidths=0.5, linecolor='white', cmap=sns.color\_palette("flare", as\_cmap=True), square=True)

#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

log\_reg = sm.Logit(y, X)#Creates model

model = log\_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

log\_reg = sm.Logit(y, X)#Recreates model

model = log\_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()

#List of removed variables

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

#Initial model confusion matrix

log\_reg\_init.pred\_table()

#Final model confusion matrix

model\_final.pred\_table()

#Predictions of final model

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

#Final model coefficients

model\_final.params

**Part V: Data Summary and Implications**

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

Where

It is clear from the equation above that Initial\_days is the most significant factor in predicting whether a patient will be readmitted by a large margin. The model reduction function may have decided that the other variables present in the final model are statistically significant, but their impact on the prediction is minimal. Despite this, there are some important facts that we can discern from the final model’s coefficients.

Regarding Initial\_days, there is a very strong positive relationship between the length of a patient’s initial stay and the likelihood that they will be readmitted. Counter to this, the coefficient of the model is large and negative, implying that a patient is very unlikely to be readmitted initially, and the risk increases tremendously with each day spent during the initial stay.

The remaining variables have relatively small coefficients and, therefore are less impactful on a patient’s readmittance chance. Patients who receive MRI or CT scans are more likely to be readmitted than patients receiving IVs or blood work as their primary service. This is likely due to the severity of diseases that require these tests for diagnosis. IVs are a common form of treatment and represent a large variety of treatment severities, delivering everything from saline for the dehydrated to morphine for gunshot-wound patients. CT scans, on the other hand, are only applicable for brain-related diagnostics that are far more detrimental to a patient’s health than dehydration. MRI scans are also used to diagnose problems within the organ or bone structures of patients, implying a heightened level of risk. Finally, blood work is a nearly compulsory procedure in hospitals, especially for patients that are being admitted. Like IVs, blood work does not imply any necessary threat to a patient’s health like a CT scan or MRI do.

There are two diagnoses with a positive coefficient, HighBlood and Stroke. HighBlood has the lowest coefficient of any predictor variable while Stroke appears to have a greater influence on readmittance. The positive coefficients of these variables imply that patients with these diagnoses are more likely to be readmitted than patients not diagnosed with these conditions.

The remaining diagnoses, Arthritis, Anxiety, and Asthma, have negative coefficients. These coefficients imply that patients diagnosed with these conditions are less likely to be readmitted than patients without these diagnoses. The likely cause of this relationship is the ability of doctors to identify the root cause of the patient’s complaint and correctly treat this underlying cause. When comparing the diagnoses with positive coefficients with those that have negative coefficients, we see that the variables with negative coefficients are more treatable than their counterparts; anxiety attacks can be treated with medication that is fast acting, asthma’s symptoms are treatable with medication and can be managed within the bounds of ordinary life, and arthritis is similarly treatable and manageable. High blood pressure, on the other hand, is treated through long-term, lifestyle changes and can have impacts on critical organs, such as the heart or brain, that cause irreparable damage or death. Strokes, similarly, cause untreatable damage and require a higher level of medical attention to manage.

Our final predictor variables, Initial\_admin and Complication\_risk, have the highest coefficients of any of the predictor variables. These variables are ordinal, with higher values representing greater medical risk or emergency. The coefficients for these variables are positive, implying that patients deemed to be at greater medical risk, or admitted under emergency circumstances, are more likely to be readmitted than patients at lower risk or admitted under there own power and will. It is not hard for us to see how a patient experiencing a medical emergency may require greater medical care and be at a higher risk of suffering another, related emergency in the near future.

1. Based on the reduced model, there are a few recommendations we can make. First, hospitals must do all they can to prioritize increasing the efficiency of care. Ensuring that patients are treated in a timely manner is the best way, by far, to reduce the number of readmissions. To do this, we recommend revisiting the prioritization of admitted patients. Patients with greater risks or experiencing emergencies should be placed at the highest level of priority, followed by patients requiring more sophisticated diagnostic techniques, such as MRIs or CT scans. Finally, it is our recommendation that it would be beneficial to overall readmission rates to provide better education and preventative care specifically concerned with high blood pressure and stroke. By education the population about the causes and preventative measures that mitigate the risk of high blood pressure and stroke, less patients will be admitted with these conditions and, potentially, lower readmission levels.

**Part VI: Demonstration**

1. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=31e3df3a-0fb6-43a2-a97c-adad016b4672>
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.

Assumptions of logistic regression. Statistics Solutions. (2021, August 11). Retrieved September 13, 2021, from <https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-logistic-regression/>.

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