**D208 Performance Assessment**

**LOGISTIC REGRESSION FOR MEDICAL DATA**

Fahim A. Akbar Student ID 001434895 Masters Data Analytics (January 1, 2021) Program Mentor: Lea Yoakem (877) 435-7948x6422 [fakbar3@wgu.edu](mailto:fakbar3@wgu.edu)

**Part I: Research Question**

**A.  Purpose of using Logistic Regression for Data Analysis**

**1.  Question**

The proposed research question is if the probability of a patient’s readmission is affected by factors such as age, gender, or present medical conditions.

**2.  Goal**

The objective of this data analysis is to predict a patient’s chance of readmission. To do so, we will test to see which independent variables in the dataset are good indicator and fit. Once we decide on which variables to use, we will perform logistic regression to determine the probability of readmission for the patients. We will also provide a logistic equation along with the statistics for the identified variables.

**Part II: Method Justification**

**B.  Logistic Regression Methods**

**1.  Logistic regression model assumptions**

According to Statology, the following assumptions must be met in order for a logistic regression model to be a good fit for the present data: (Zach 2020)

* The dependent variable is binary
* Observations are independent. Residual plots against time can be used to check if there is a random pattern in the order of observations. If there isn’t, then the observations do not satisfy this assumption
* No multicollinearity should be present among the variables.
* No extreme outliers should be present.
* Explains the relationship between one dependent binary variable and one or more independent nominal variables
* A linear relationship should be present between the independent variables and logit of the dependent variable. Logit is defined as Logit(p) = log(p / (1-p)), where p is the probability of a positive outcome.
* The dataset offers a large sample size. Our data set has over 10,000 entries and is therefore sufficient.

**2.  Benefits of using Python**

In this assessment, Python will be used for the logistic regression model of the medical data. Python provides several packages that allow for statistical analyses, such as Pandas, Scipy and Statsmodels. Python enables easy-to-understand visualizations of variables and the observations within them. It allows you to analyze data easily into separate groups without modifying the original dataset. Python also has commands specific to logistic regression, including LogisticRegression, model.fit, and model.coef\_. These commands will help with performing the regression analyses and visualizing them.

**3. Why logistic regression is an appropriate technique to analyze the research question**

For this assessment we are not trying to predict specific values. Instead, our goal is to determine the probability of an event occurring. The event we’re interested in predicting is the chance of a patient being readmitted to the hospital. Logistic regression is an appropriate technique to analyze the probability of patient readmission. The patient readmission is a binary predictor to a categorical variable (Readmission = Yes or No.) The patient observations are the independent variables of the data set that will be tested.

**Part III: Data Preparation**

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

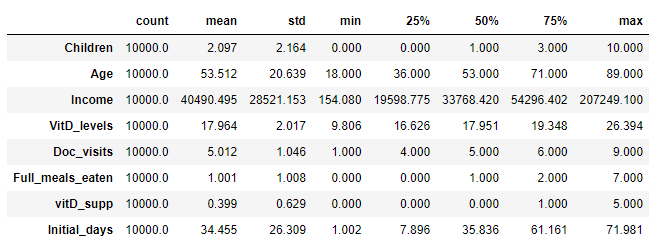
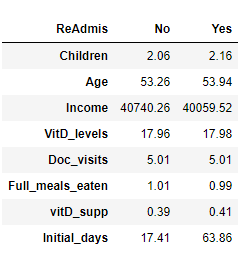
**1. Relevant data preprocessing goal and data manipulations**

To prepare the data for logistic regression, we will first need to address null and missing values. We will either change the null values to zeros or populate them based on averages. Continuous variables will have histograms created to see if they have too many categories, and if we need combine categories. For example, we’ll condense the Age variable by groups of 10 instead of five years. We will need to determine the ranges for other variables like HighBlood.

**2. Summary statistics**

P-values and coefficients for the independent variables will be identified to figure out which independent variables impact the probability and dependent variable. For this analysis, the target dependent variable is ReAdmis. This variable represents whether a patient has been readmitted to the hospital. We believe that performing logistic regression will help predict which patients have a higher chance of being readmitted. This analysis could help the hospital plan to reduce readmissions, saving time and resources for the hospital and the patients. For this analysis, the following predictor variables will be used: Age, Gender, VitD\_levels, HighBlood, Stroke, Complication, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic\_rhinitis, Reflux\_esophagitis, and Asthma.

The tables below the numeric variables with their summary statistics. We found that the mean Age and VitD\_levels between patients readmitted and those that were not are near equal. We also found a a significant difference in the mean Initial\_days of patients that were not readmitted compared to those that were. We also see in the table the standard deviations of each numerical variable as well as the dispersion in the interquartile ranges. The variables Age, VitD\_Levels, Doc\_visits, and Initial\_days are normally distributed. The variables Children, Full\_meals\_eaten, Income, and vitD\_supp are not normally distributed variables.



The categorical variables are visualized in section 4 of part III. The statistics summary for the catergorical variables show that Reflux\_esophagitis, BackPain, are normally distributed. Gender was also normally distributed outside of the Nonbinary responses. On the contrary, the responses for Marital, Soft\_drink, Initial\_admin, Stroke, Complication\_risk, and Diabetes are separated by large margins. After comparing the variables against ReAdmis, it was found that Marital and Initial\_admin are normally distributed while Stroke and Diabetes are not. Furthermore, there were more No responses to ReAdmis than Yes responses, which confirms the expected distribution of readmission rates. Ideally hospitals do their best to ensure that readmission rates are low, so this data falls in line with what’s expected.

**PART III**

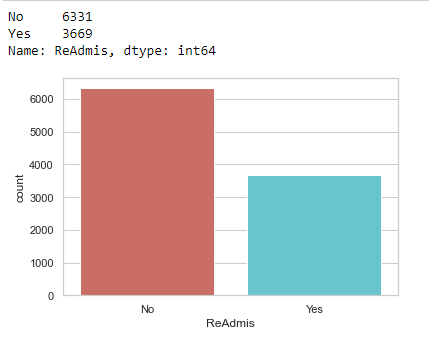
**3. Data Preparation Steps**

The first step in preparing the data is to make sure that there are no missing data entries in any of the columns. Next, we will ensure that none of the data in the columns is duplicated. We will make sure that none of the columns or rows are duplicated, to further prevent dealing with repeated entries. For the logistics analysis, several columns in the dataset were deemed irrelevant and were subsequently dropped from the dataset (i.e latitude, longitude). Additionally, the predictor variables will need to be scaled, and the data will need to split into test and training sets. After the predictor variables have been scaled, the “yes/no” entries for the categorical variables will need to be converted to 1 and 0, respectively. The marriage variable will also be condensed to fewer unique answers. Once all the necessary modifications to the data set are made, the columns that will be used for the logistics regression will be identified and split into categorical or numerical variable types.

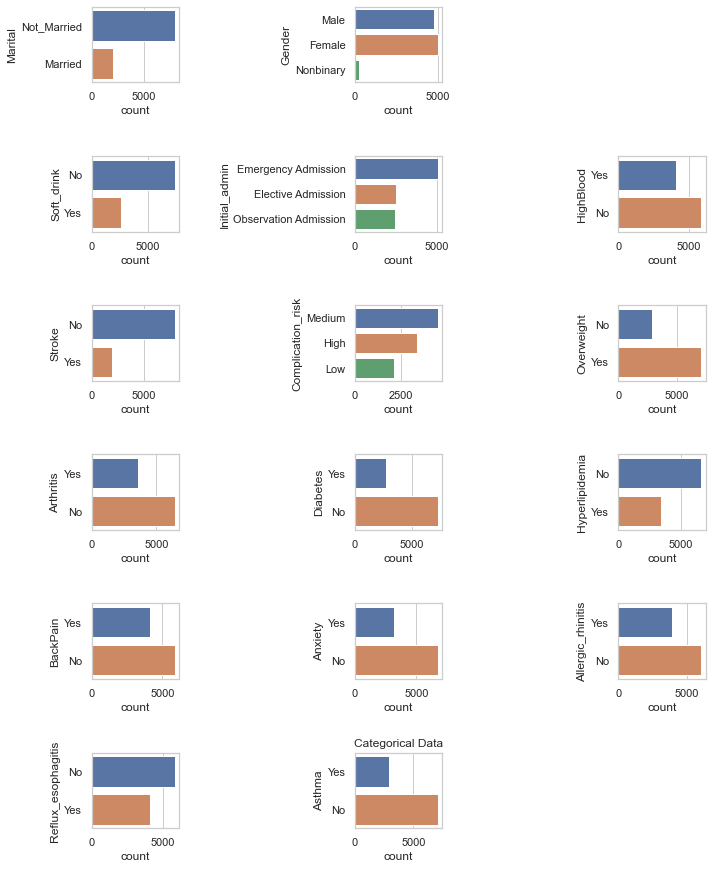
**Code used for preparing data:**

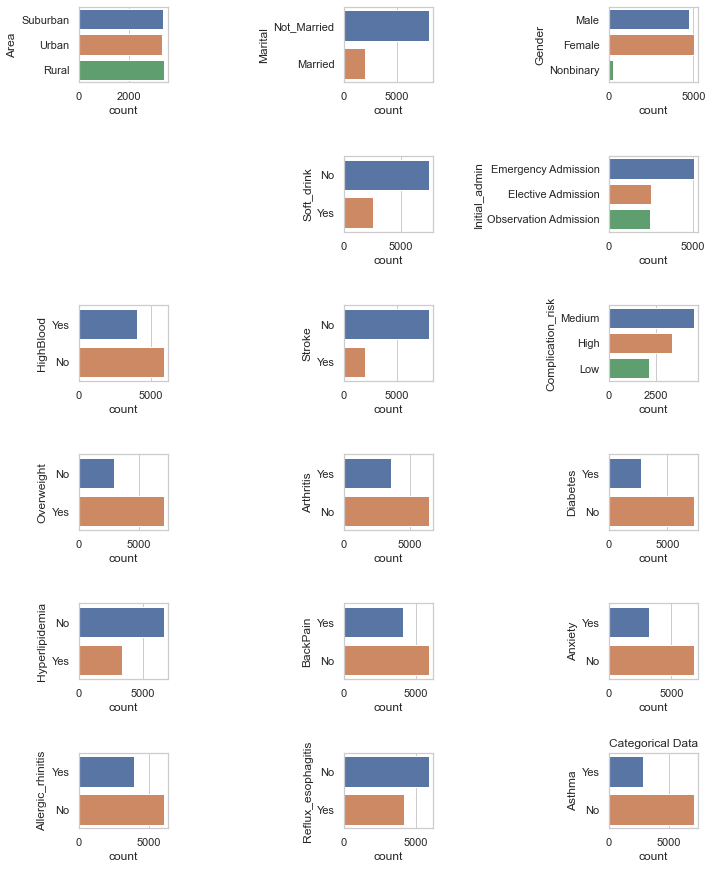
**4. Univariate and bivariate visualizations of the distributions of variables**

The cleaned data set was used to prepare visualizations of the variables. These will help determine which variables will be used in the logistic regression model. First we observed the target variable.

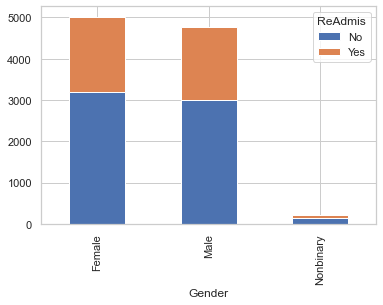
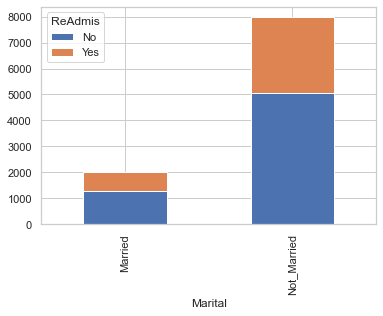


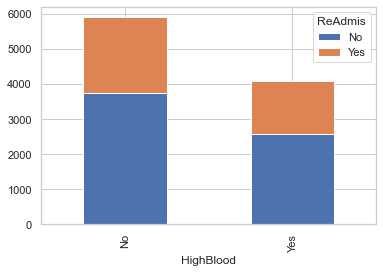
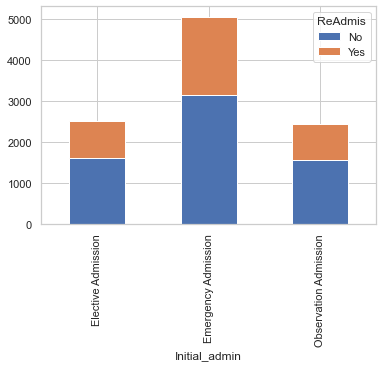
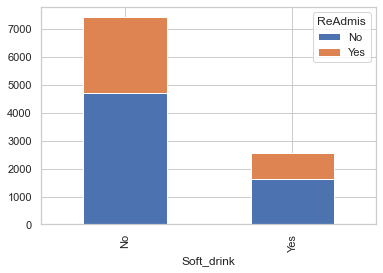
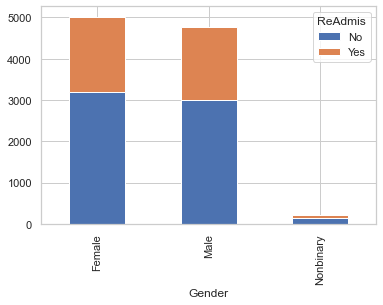
Here we see that the majority of patients were not readmitted. Next, we review the independent variables, starting with visualizing the categorical data.

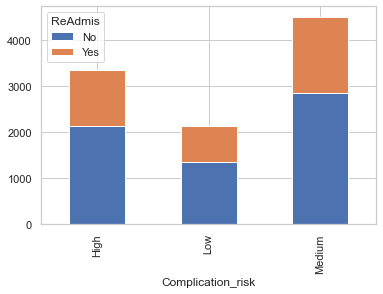
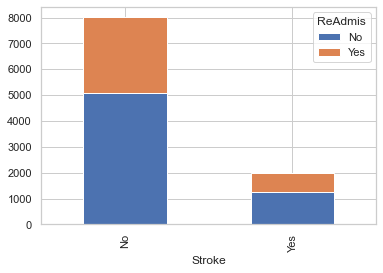


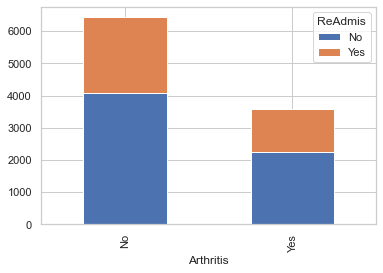
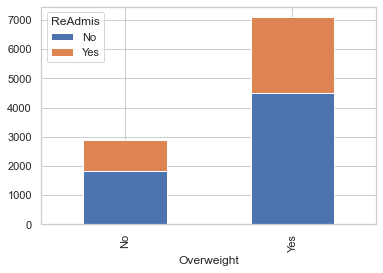


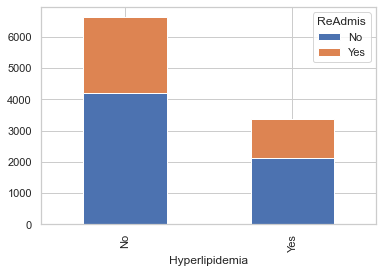
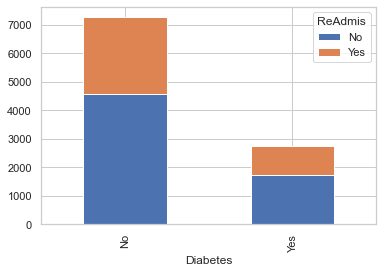
For bivariate visualizations, we compare the ReAdmis variable with the identified categorical variables.

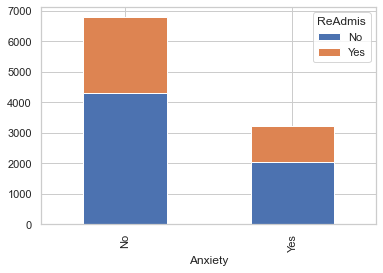
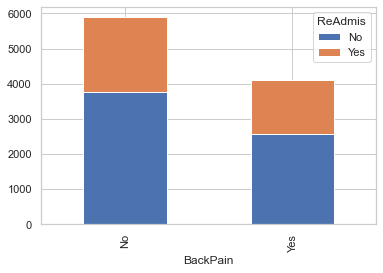


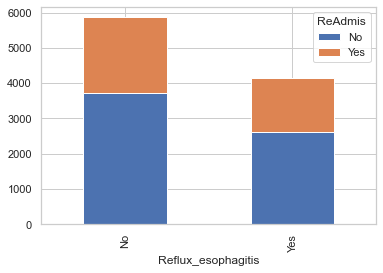
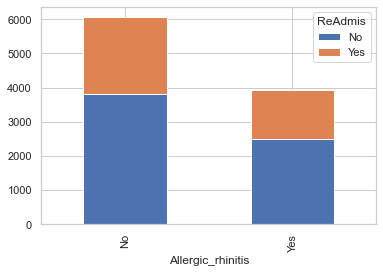


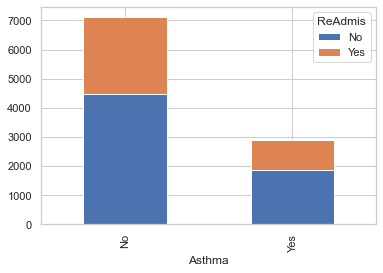


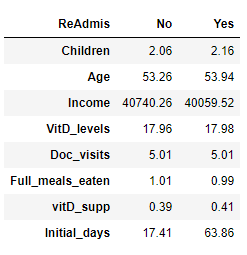




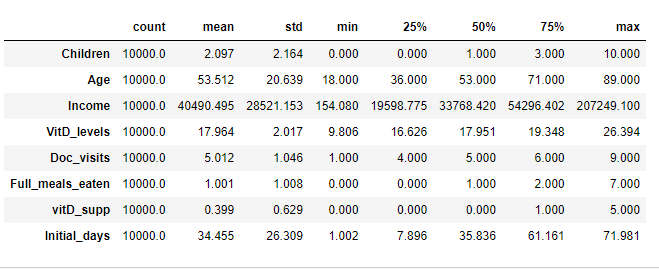




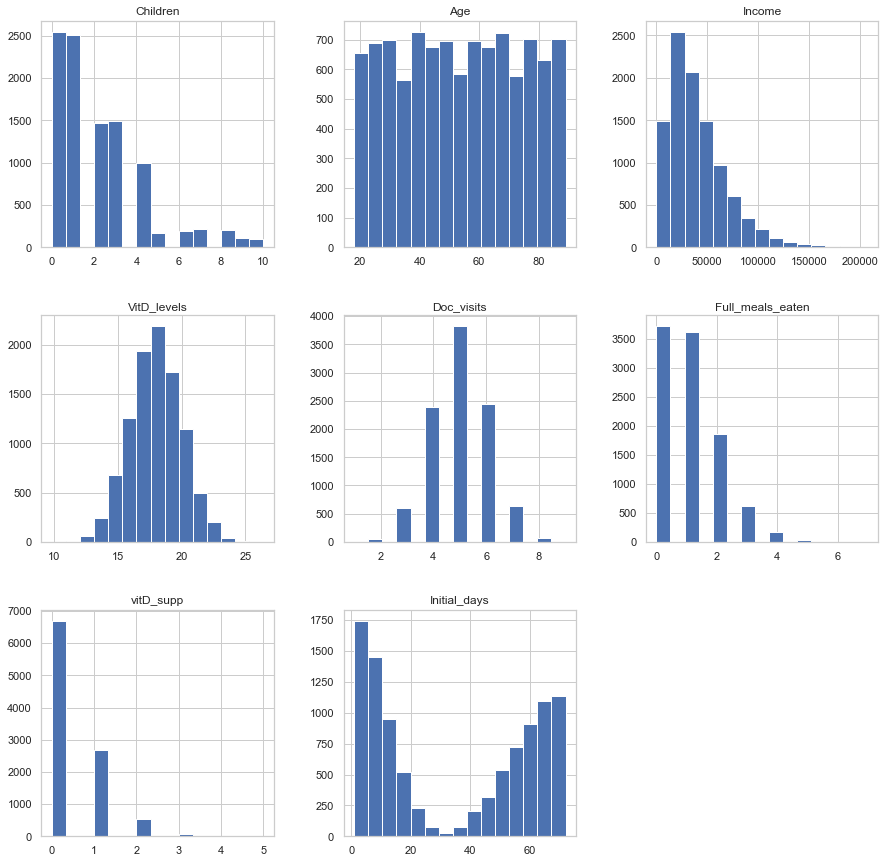


Using these results, we conclude that all of the variables could be good predictors of the outcome. After selecting the variables to use, they’ll be converted to binary outcomes for the regression. Our new target variable will be ReAdmis\_Yes, and we’ll test to find the probability of an independent variable impacting this target variable. The numerical data is reviewed to determine the means for the variables within the Yes and No of the ReAdmis target variable. 

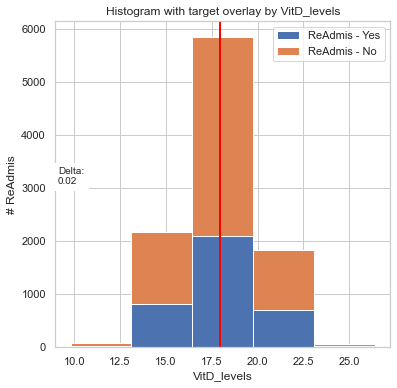
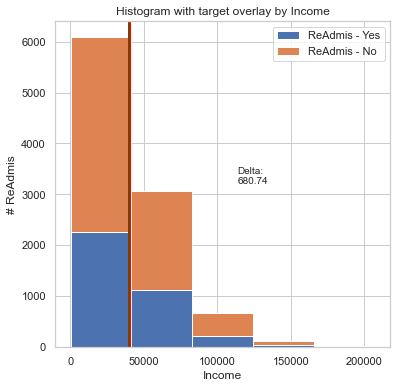
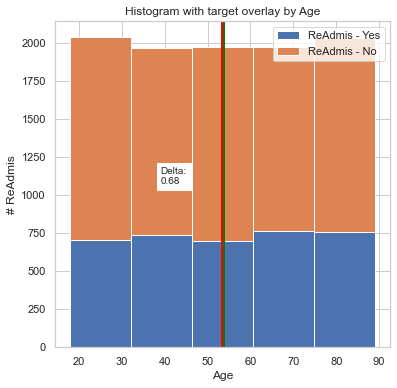
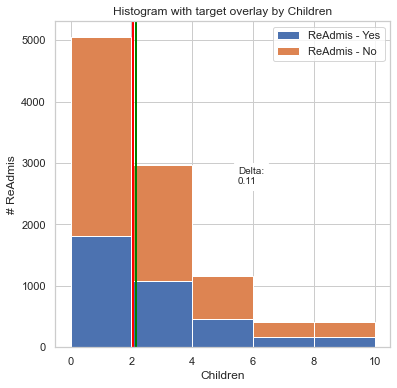
The table above shows that the means for Children, Age, VidD\_levels, Doc\_Visits, Full\_meals\_eaten and vidD\_supp are statistically equal, with little difference between them. We also found patients who were not readmitted had marginally higher income levels than patients who were. This could be a contributing factor to readmission rate, and we’ll proceed keeping this in mind. We also saw that the Initial\_days spent at the hospital were significantly lower for patients not readmitted. The table below outlines the statistics for the numerical data:

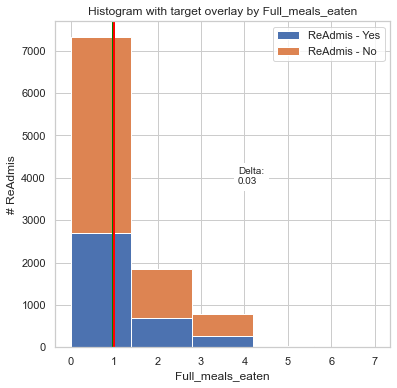
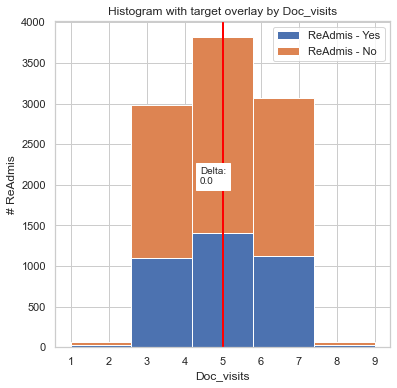


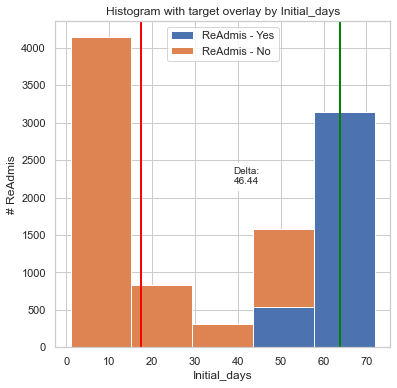
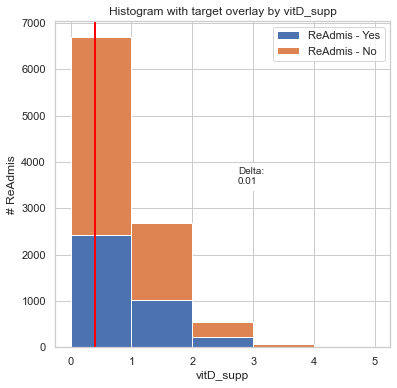
Now we’ll identify which numerical variables have a high probability of impacting the target variable. The histograms below were created and used to visualize the numeric data.



In addition, we plotted the numerical data and compared it against the ReAdmis variable. By plotting the delta between the two, the responses will help us further narrow down which variables are ideal to use as good predictors.





After determining the means and delta for the numerical data, the data for patients who were not readmitted was removed. That way, the data represented will only pertain to patients who were readmitted and allow us to create a prediction model.

**Code used for cleaning**

# import packages and data set that will be used for the logistics regression analysis

import seaborn as sb

sb.set(style="white")

sb.set(style="whitegrid", color\_codes=True)

import sklearn

from sklearn.metrics import confusion\_matrix

from sklearn import preprocessing

from sklearn.decomposition import PCA

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

from sklearn.metrics import classification\_report

from sklearn import metrics

import matplotlib.pyplot as plt

plt.rc("font", size=14)

import numpy as np

import scipy.stats as stats

import statsmodels.api as sm

import statsmodels.formula.api as smf

from IPython.core.display import HTML

from IPython.display import display

import pandas as pd

from pandas import Series, DataFrame

from sklearn.metrics import classification\_report, confusion\_matrix

from imblearn.over\_sampling import SMOTE

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

df = pd.read\_csv (r'C:\Users\fahim\Documents\0\_WGUDocuments\d208\1medical\_clean.csv')

# check if there is any missing data entries. Output should be false.

df.isna().any()

# check if there are any duplicate data in columns

df[df.duplicated()]

# check if any of the columns are duplicated. Output should be False.

df.columns.duplicated().any()

# to begin cleaning the data, first drop the columns for survey questions.

df.drop(['Item8','Item7','Item6','Item5','Item4','Item3','Item2','Item1'],axis = 1,inplace=True)

# drop all of the columns for demographic data

df.drop(['City','State','County','Area','Zip','Lat','Lng','Population','TimeZone'],axis = 1,inplace=True)

# drop the remaining columns for other data not needed in this analysis

df.drop(['Customer\_id','Job','CaseOrder'],axis = 1, inplace=True)

# verify that all the columns were dropped before proceeding

df.info()

# Callout the target variable "ReAdmis" and describe it's characteristics

target = 'ReAdmis'

for idx, c in enumerate(df.loc[:, df.columns == target]):

if df.dtypes[c] == "object":

print('\n{}. {} is categorical: {}.'.format(idx+1,c,df[c].unique()))

for idx,name in enumerate(df[c].value\_counts().index.tolist()):

print('\t{:<8}:{:>6}'.format(name,df[c].value\_counts()[idx]))

print('\n{}. {} is numerical.'.format(idx+1, c))

# consolidate the data for the marital variable

df['Marital']=np.where(df['Marital'] =='Widowed', 'Not\_Married',df['Marital'])

df['Marital']=np.where(df['Marital'] =='Separated', 'Not\_Married',df['Marital'])

df['Marital']=np.where(df['Marital'] =='Never Married', 'Not\_Married',df['Marital'])

df['Marital']=np.where(df['Marital'] =='Divorced', 'Not\_Married',df['Marital'])

# show the new values for the marital variable to make sure the consolidation worked

df['Marital'].unique()

# Output and describe the input variables of the data set

for idx, c in enumerate(df.loc[:, df.columns != target]):

if df.dtypes[c] == "object":

print('\n{}. {} is categorical: {}.'.format(idx+1,c,df[c].unique()))

for idx,name in enumerate(df[c].value\_counts().index.tolist()):

print('\t{:<20}:{:>6}'.format(name,df[c].value\_counts()[idx]))

print('{}'.format(df[c].describe()))

print('\n{}. {} is numerical.'.format(idx+1, c))

# print the columns for categorical data

CategoricalData = df.select\_dtypes(include = "object").columns

print(CategoricalData)

# print the columns for numerical data

NumericalData = df.select\_dtypes(include = "number").columns

print (NumericalData)

**5.  Provide a copy of the prepared data set.**

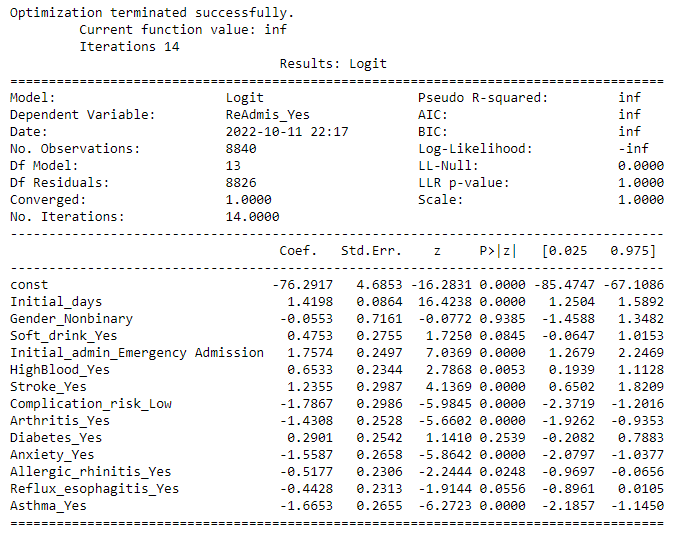
A copy of the cleaned data set, titled “1medical\_clean-PREPAREDTASK2” is provided in the task submission.

**Part IV: Model Comparison and Analysis**

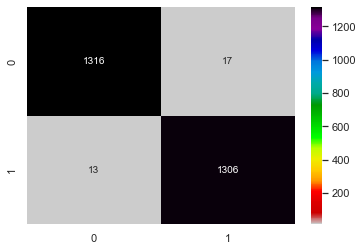
**D.  Comparison an initial and a reduced logistic regression**

**1. Initial logistic regression model**

Logit is used to run initial model. The output provided us with the P-value, coefficients, standard error values, and Z-value of the variables.

****

To better visualize the data, we also created this confusion matrix:

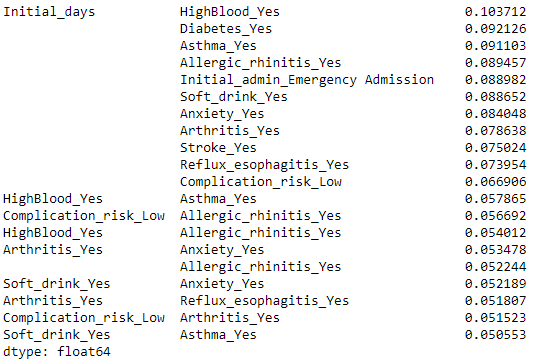


**2.  Justify a statistically based variable selection procedure and a model evaluation metric to reduce the initial model in a way that aligns with the research question**.

The number and percent of predictions can be shown to provide a confidence level in the initial model.



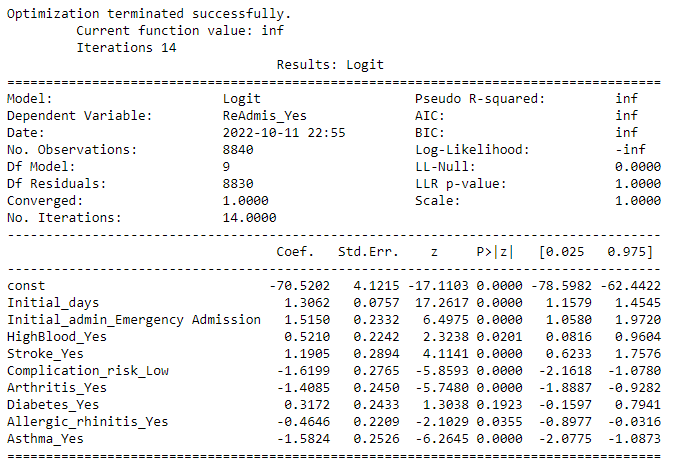
A correlation matrix with the predictors was then created to compare the potential pairs of variables.



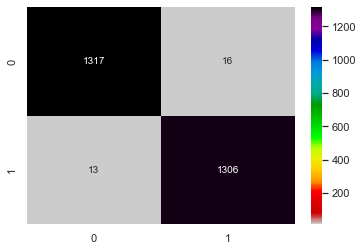
The initial output and the correlation matrix list helped us determine which variables to remove before creating the reduced model. Based on the assumptions of logistic regression, this reduced model will not contain variables that have high P-values or high multi-collinearity. Therefore, we removed the following variables due to their high P-values: Gender\_Nonbinary, Soft\_drink\_yes, and Reflux\_esophagitis\_Yes.

**3.  Reduced logistic regression model.**

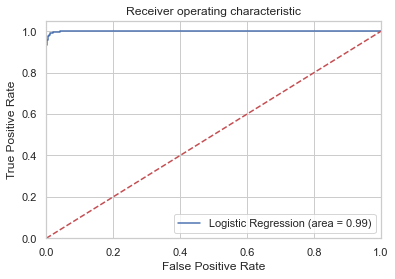
Now that we have our reduced model, we can proceed and run Logit again.



The new model provided us with this new confusion matrix and prediction percentage:



Using the reduced model and variables, we built a Receiver Operating Characteristic (ROC) graph. This linear slope is that of a random classifier, while the blue curve indicates the true positive rate of the classifiers used. If the blue arc is closer to the top left, then variables used have generated a greater confidence for the model. Because this is what we see with the graph produced, we can confirm our results with confidence.



**Code used to support the implementation of the logistic regression models.**

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

The full code for the project is provided at the end section **Part VI: Demonstration.** Furthermore, a pdf print of the Jupyter notebook used for running the python scripts is attached in task submission.

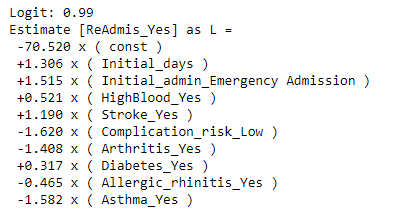
**Part V: Data Summary and Implications**

**F. Summary of findings and assumptions**

**Data analysis results:**

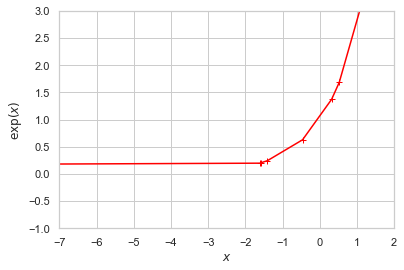
**Regression equation for the reduced model:**

After conducting the analysis, we created the following logistic regression equation for the reduced model:



**Interpretation of coefficients of the statistically significant variables of the model:**

The regression coefficient was provided by the output, and it outlines the change in the response variable whenever we increased the predictor by a value of one. We plot this data so that the results are easier to visualize and interpret.



In this chart, each of variables are listed as the X’s on the line. A higher chance of readmission correlates with the positive coefficients, while a lower chance correlates with the negative coefficients. Based on our analysis, the following variables are linked to a patient’s higher chance of readmission: Initial\_days, Initial\_admin\_Emergency Admission, and Stroke\_Yes. The remaining variables indicate a lower likelihood, but all variables used to create this model can potentially influence a patient’s chance of being readmitted.

**Statistical and Practical significance of the model:**

The hospital can use our analysis to help evaluate and treat patients that have either been admitted for an emergency or have had a stroke. These were strong predictor variables for a patient’s chance of being readmitted, so now the hospital could plan more efficient protocols involving these kinds of patients. They can also plan protocols based on the other strong predictor variables listed. By doing so, they could decrease the likelihood of patient readmission which will in turn help the hospital save time and resources. This can also benefit the patients because giving them the best treatment ahead of time based on their conditions will help to further reduce their chances of being readmitted. If any patient shows a positive result for any of the predictor variables, the hospital could plan ahead and anticipate their readmission in case preventive care or protocols aren’t enough. A patient who shows a combination of these variables would in turn have a higher chance of being readmitted. Knowing this could help the hospital further determine which patients are at the highest risk and they can plan treatments accordingly.

**Data analysis limitations:**

The first limitation we address is that the predictor variables are limited to high level information. They do not provide a diagnosis or solution to the higher readmission rates. Rather, they can be used to help predict if a patient has a high chance of readmission. These results are purely predictive and must be treated as such. Furthermore, the analysis is limited by the data set provided, as there could be additional variables that were not accounted for during the surveys. These could range from genetic history, recent exposure to certain viruses, additional health complications, and so on. The data provided is what the hospital deemed most important, so we are limited to work within that given framework. Lastly, as with any logistic regression, there is the potential of the model being overfit.

**Recommended course of action**

Some of the determined predictor variables are beyond the control of both the patient and the hospital. Emergency situations and strokes are near impossible to predict for ahead of time, so the best course of action in those circumstances would be to improve upon existing protocol whenever possible. Improving the treatment process is the best thing to help The number of initial days is a variable that can be reduced in order to decrease the likelihood of readmission. By making sure the patient receives the most accurate diagnosis and procedures as soon as possible, they in turn won’t have to spend too much time at the hospital getting treated. To address concerns of the model being overfit, we could run the regression multiple times or having additional analysts test for different variables could be beneficial to increase overall confidence.

**Part VI: Demonstration**

**Link to the Panopto Video recording:**

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=bd7cb712-24d5-44eb-9e21-af2c0055a5ef>

**Sources for third party code:**

**Code for importing packages, preparing data, and performing logistic regression:**

<https://www.kaggle.com/code/prashant111/logistic-regression-classifier-tutorial/notebook>

<https://www.kaggle.com/code/miguelrodriguezolmos/plot-the-logistic-map-with-python-matplotlib/notebook>

<https://www.kaggle.com/code/vipulgandhi/linear-regression/notebook>

<https://www.kaggle.com/code/alexnystrom/hospital-data-logistic-regression>

<https://www.kaggle.com/code/neisha/heart-disease-prediction-using-logistic-regression>

**References**

Zach. (2020, October 13). The 6 Assumptions of Logistic Regression (With Examples). Statology. Statology.org/assumptions-of-logistic-regression/

**Full Code used for this Project**

# import packages and data set that will be used for the logistics regression analysis

import seaborn as sb

sb.set(style="white")

sb.set(style="whitegrid", color\_codes=True)

import sklearn

from sklearn.metrics import confusion\_matrix

from sklearn import preprocessing

from sklearn.decomposition import PCA

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

from sklearn.metrics import classification\_report

from sklearn import metrics

import matplotlib.pyplot as plt

plt.rc("font", size=14)

import numpy as np

import scipy.stats as stats

import statsmodels.api as sm

import statsmodels.formula.api as smf

from IPython.core.display import HTML

from IPython.display import display

import pandas as pd

from pandas import Series, DataFrame

from sklearn.metrics import classification\_report, confusion\_matrix

from imblearn.over\_sampling import SMOTE

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

df = pd.read\_csv (r'C:\Users\fahim\Documents\0\_WGUDocuments\d208\1medical\_clean.csv')

# check if there is any missing data entries. Output should be false.

df.isna().any()

# check if there are any duplicate data in columns

df[df.duplicated()]

# check if any of the columns are duplicated. Output should be False.

df.columns.duplicated().any()

# to begin cleaning the data, first drop the columns for survey questions.

df.drop(['Item8','Item7','Item6','Item5','Item4','Item3','Item2','Item1'],axis = 1,inplace=True)

# drop all of the columns for demographic data

df.drop(['City','State','County','Area','Zip','Lat','Lng','Population','TimeZone'],axis = 1,inplace=True)

# drop the remaining columns for other data not needed in this analysis

df.drop(['Customer\_id','Job','CaseOrder'],axis = 1, inplace=True)

# verify that all the columns were dropped before proceeding

df.info()

# Callout the target variable "ReAdmis" and describe it's characteristics

target = 'ReAdmis'

for idx, c in enumerate(df.loc[:, df.columns == target]):

if df.dtypes[c] == "object":

print('\n{}. {} is categorical: {}.'.format(idx+1,c,df[c].unique()))

for idx,name in enumerate(df[c].value\_counts().index.tolist()):

print('\t{:<8}:{:>6}'.format(name,df[c].value\_counts()[idx]))

print('\n{}. {} is numerical.'.format(idx+1, c))

# consolidate the data for the marital variable

df['Marital']=np.where(df['Marital'] =='Widowed', 'Not\_Married',df['Marital'])

df['Marital']=np.where(df['Marital'] =='Separated', 'Not\_Married',df['Marital'])

df['Marital']=np.where(df['Marital'] =='Never Married', 'Not\_Married',df['Marital'])

df['Marital']=np.where(df['Marital'] =='Divorced', 'Not\_Married',df['Marital'])

# show the new values for the marital variable to make sure the consolidation worked

df['Marital'].unique()

# Output and describe the input variables of the data set

for idx, c in enumerate(df.loc[:, df.columns != target]):

if df.dtypes[c] == "object":

print('\n{}. {} is categorical: {}.'.format(idx+1,c,df[c].unique()))

for idx,name in enumerate(df[c].value\_counts().index.tolist()):

print('\t{:<20}:{:>6}'.format(name,df[c].value\_counts()[idx]))

print('{}'.format(df[c].describe()))

print('\n{}. {} is numerical.'.format(idx+1, c))

# print the columns for categorical data

CategoricalData = df.select\_dtypes(include = "object").columns

print(CategoricalData)

# print the columns for numerical data

NumericalData = df.select\_dtypes(include = "number").columns

print (NumericalData)

# set up the "get redundant pairs" parameter for this analysis

def get\_redundant\_pairs(df):

'''Get diagonal and lower triangular pairs of correlation matrix'''

pairs\_to\_drop = set()

cols = df.columns

for i in range(0, df.shape[1]):

for j in range(0, i+1):

pairs\_to\_drop.add((cols[i], cols[j]))

return pairs\_to\_drop

# set up the "custom correlation matrix" parameters for this analysis

def custom\_corr\_matrix(df, title):

fig = plt.figure(figsize=(30, 30))

sns.set(font\_scale=1.0)

sns.heatmap(data=df.corr().round(1), annot=True,annot\_kws={'size':30})

print(get\_top\_abs\_correlations(df))

#plt.savefig('output/' + COURSE + '/fig\_corr\_matrix\_' + title + '.png', facecolor='w')

plt.show()

def get\_top\_abs\_correlations(df, n=5):

au\_corr = df.corr().abs().unstack()

labels\_to\_drop = get\_redundant\_pairs(df)

au\_corr = au\_corr.drop(labels=labels\_to\_drop).sort\_values(ascending=False)

return au\_corr[0:n]

# establish the parameters for plotting histograms

def plot\_histogram(c):

df\_yes = df[df.ReAdmis\_Yes==1][c]

df\_no = df[df.ReAdmis\_Yes==0][c]

yes\_mean = df\_yes.mean();

no\_mean = df\_no.mean();

fig,ax = plt.subplots(figsize=(6,6))

ax.hist([df\_yes,df\_no], bins=5, stacked=True)

ax.legend(['ReAdmis - Yes','ReAdmis - No'])

ymin, ymax = ax.get\_ylim();

xmin, xmax = ax.get\_xlim()

ax.axvline(yes\_mean, color='green', lw=2) # yes mean

ax.axvline(no\_mean, color='red', lw=2) # no mean

ax.text((xmax-xmin)/2, (ymax-ymin)/2, 'Delta:\n' + str(round(abs(yes\_mean - no\_mean),2)),

bbox={'facecolor':'white'})

plt.title('Histogram with target overlay by ' + str(c))

plt.xlabel(c);

plt.ylabel('# ReAdmis');

plt.show();

# set up a helper function to plot grouped bar plot

def plot\_stacked(c):

df.groupby([c,target]).size().unstack().plot(kind='bar', stacked=True)

# data cleaning is complete and parameters have been set, and we can now proceed to the analysis

# begin visuallizing the data and include the univariate and bivariate analyses

print(df[target].value\_counts())

sb.countplot(x=target, data=df, palette='hls')

plt.show()

# visualize the categorical data and include the univariate and bivariate analyses

fig = plt.figure(figsize=(10, 20))

for i, col in enumerate(CategoricalData):

if col != target:

plt.subplot(10, 3, i+1)

ax = sb.countplot(y=col, data=df)

fig.tight\_layout(h\_pad=4, w\_pad=4)

plt.title('Categorical Data')

plt.show()

# Compare the target variable "ReAdmis" to ALL of the categorical variables

tbl = pd.pivot\_table(df.groupby(['Marital','ReAdmis']).size().reset\_index(),

values=0,

index='Marital',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Gender','ReAdmis']).size().reset\_index(),

values=0,

index='Gender',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Soft\_drink','ReAdmis']).size().reset\_index(),

values=0,

index='Soft\_drink',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Initial\_admin','ReAdmis']).size().reset\_index(),

values=0,

index='Initial\_admin',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['HighBlood','ReAdmis']).size().reset\_index(),

values=0,

index='HighBlood',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Stroke','ReAdmis']).size().reset\_index(),

values=0,

index='Stroke',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Complication\_risk','ReAdmis']).size().reset\_index(),

values=0,

index='Complication\_risk',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Overweight','ReAdmis']).size().reset\_index(),

values=0,

index='Overweight',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Arthritis','ReAdmis']).size().reset\_index(),

values=0,

index='Arthritis',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Diabetes','ReAdmis']).size().reset\_index(),

values=0,

index='Diabetes',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Hyperlipidemia','ReAdmis']).size().reset\_index(),

values=0,

index='Hyperlipidemia',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['BackPain','ReAdmis']).size().reset\_index(),

values=0,

index='BackPain',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Anxiety','ReAdmis']).size().reset\_index(),

values=0,

index='Anxiety',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Allergic\_rhinitis','ReAdmis']).size().reset\_index(),

values=0,

index='Allergic\_rhinitis',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Reflux\_esophagitis','ReAdmis']).size().reset\_index(),

values=0,

index='Reflux\_esophagitis',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

tbl = pd.pivot\_table(df.groupby(['Asthma','ReAdmis']).size().reset\_index(),

values=0,

index='Asthma',

columns=['ReAdmis'],

aggfunc=np.sum)

tbl.plot(kind='bar',stacked=True)

# review numerical mean data compared to target variable

df.groupby(target).mean().round(2).T

# retrieve statistics for numerical data

df[NumericalData].describe().round(3).T

# create histogram plots for the numeric data

fig = plt.figure(figsize=(10, 20))

ax = df[NumericalData].hist(bins = 15, figsize=(15,15))

plt.title('Numeric Data')

fig.tight\_layout(h\_pad=5, w\_pad=5)

plt.show()

# update the target variable to "Yes" for readmissions

target = 'ReAdmis\_Yes'

# convert the categorical data into binary data for the analysis

for c in CategoricalData:

if c in df.columns:

df = pd.get\_dummies(df, columns=[c], drop\_first=True)

pred\_vars = df.select\_dtypes(include="uint8").columns.tolist()

print(pred\_vars)

# create histograms with overlay of the target variable

plot\_histogram('Children')

plot\_histogram('Age')

plot\_histogram('Income')

plot\_histogram('VitD\_levels')

plot\_histogram('Doc\_visits')

plot\_histogram('Full\_meals\_eaten')

plot\_histogram('vitD\_supp')

plot\_histogram('Initial\_days')

# since our target variable is the "yes" responses, drop unwanted "no" data

cols\_to\_be\_removed = ['ReAdmis\_No']

# print a list of dropped "no" data

print('data to be removed: {}'.format(cols\_to\_be\_removed))

# loop through list to see if it is in current dataframe and if the column needs to be dropped

for c in cols\_to\_be\_removed:

if c in df.columns:

df.drop(columns = c, inplace=True)

print('Data named [{}] has been removed.'.format(c))

# rebalance the prepared data using the SMOTE oversample function

os = SMOTE(random\_state=0)

X = df.loc[:, df.columns != 'ReAdmis\_Yes']

y = df.loc[:, df.columns == 'ReAdmis\_Yes']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,test\_size=0.3, random\_state=0)

columns = X\_train.columns

os\_data\_X,os\_data\_y=os.fit\_resample(X\_train, y\_train)

os\_data\_X = pd.DataFrame(data=os\_data\_X,columns=columns )

os\_data\_y= pd.DataFrame(data=os\_data\_y,columns=['ReAdmis\_Yes'])

# Check the numbers of our data to see if new dataset is now balanced after traning and sampling

print("length of oversampled data is ",len(os\_data\_X))

print("Number of no ReAdmis in oversampled data",len(os\_data\_y[os\_data\_y['ReAdmis\_Yes']==0]))

print("Number of ReAdmis",len(os\_data\_y[os\_data\_y['ReAdmis\_Yes']==1]))

print("Proportion of no ReAdmis data in oversampled data is",len(os\_data\_y[os\_data\_y['ReAdmis\_Yes']==0])/len(os\_data\_X))

print("Proportion of ReAdmis data in oversampled data is",len(os\_data\_y[os\_data\_y['ReAdmis\_Yes']==1])/len(os\_data\_X))

# implement the RFE feature reduction

data\_final\_vars=df.columns.values.tolist()

y=[target]

X=[i for i in data\_final\_vars if i not in y]

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

rfe = RFE(logreg)

rfe = rfe.fit(X\_train, y\_train)

print(rfe.support\_)

print(rfe.ranking\_)

X\_train.columns[rfe.support\_]

# create the initial model for the logisitics regression

X=os\_data\_X[features] # from RFE above

Xc = sm.add\_constant(X) # reset

y=os\_data\_y[target]

logit\_model=sm.Logit(y,Xc)

result=logit\_model.fit()

print(result.summary2())

# create a confusion matrix for the initial model

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,test\_size=0.3, random\_state=0)

lgr = LogisticRegression()

lgr.fit(X\_train, y\_train)

predicted = lgr.predict(X\_test)

expected = y\_test

confusion = pd.DataFrame(confusion\_matrix(y\_true=expected, y\_pred=predicted),index=range(2),columns=range(2))

axes = sb.heatmap(confusion, annot=True,cmap='nipy\_spectral\_r', fmt='g')

# calculate number and percent of predictions

correct = sum(np.diagonal(confusion)) # on diag

total = confusion.values.sum()

incorrect = total - correct # off diag

print('Correct predictions on diagonal: {} ({:.0%})'.format( correct, correct / total ))

print('Incorrect predictions off diagonal: {} ({:.0%})'.format( incorrect, incorrect / total ))

# find predictor pairs with high coorelation

get\_top\_abs\_correlations(X, 20)

# reduce the model before running the logistic regression

features.remove('Soft\_drink\_Yes') # high p-value

# reduce the model before running the logistic regression

features.remove('Reflux\_esophagitis\_Yes') # high p-value

# create the reduced logit model

X=os\_data\_X[features]

y=os\_data\_y[target]

Xc = sm.add\_constant(X) # reset

logit\_model=sm.Logit(y,Xc)

result=logit\_model.fit()

print(result.summary2())

# confustion matrix for final model

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,test\_size=0.3, random\_state=0)

lgr = LogisticRegression()

lgr.fit(X\_train, y\_train)

predicted = lgr.predict(X\_test)

expected = y\_test

confusion = pd.DataFrame(confusion\_matrix(y\_true=expected, y\_pred=predicted),index=range(2),columns=range(2))

axes = sb.heatmap(confusion, annot=True,cmap='nipy\_spectral\_r', fmt='g')

# calculate number and percent of predictions for the reduced model

correct = sum(np.diagonal(confusion)) # on diag

total = confusion.values.sum()

incorrect = total - correct # off diag

print('Correct predictions on diagonal: {} ({:.0%})'.format( correct, correct / total ))

print('Incorrect predictions off diagonal: {} ({:.0%})'.format( incorrect, incorrect / total ))

# classification report for the reduced model

print(classification\_report(expected, predicted))

# plot ROC Curve

logit\_roc\_auc = roc\_auc\_score(y\_test, lgr.predict(X\_test))

fpr, tpr, thresholds = roc\_curve(y\_test, lgr.predict\_proba(X\_test)

[:,1])

plt.figure()

plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %logit\_roc\_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="lower right")

plt.savefig('Log\_ROC')

plt.show()

# Discuss Results of the reduced logistic regression model

# create an equation of the regression

print('Logit: {:.2f}'.format(logit\_roc\_auc))

equation = result.summary2().tables[1]

print('Estimate [{}] as L = '.format(result.summary2().tables[0][1][1]))

for i in equation.itertuples():

print(' {:+.3f} x ( {} ) '.format(i[1],i[0]))

# create a visualization of the logistic coefficient

import matplotlib.pyplot as plt

X\_coeff = []

for i in equation.itertuples():

X\_coeff.append(i[1])

X\_coeff.sort()

x = X\_coeff

y = np.exp(x)

plt.figure()

plt.plot(X\_coeff, y, color="red", marker="+")

plt.xlim([-7.0, 2.0])

plt.ylim([-1.0, 3.0])

plt.xlabel('$x$')

plt.ylabel('$\exp(x)$')

plt.show()

# export the prepared dataset used

df.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d208\1medical\_clean-PREPAREDTASK2.csv')