#Import the libraries needed

from pandas import Series, DataFrame

from pylab import rcParams

from scipy.stats import spearmanr

from sklearn import datasets

from sklearn import metrics

from sklearn import preprocessing

from sklearn.linear\_model import LogisticRegression

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

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

from sklearn.preprocessing import scale

from statsmodels.formula.api import ols

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import scipy

import seaborn as sns

import sklearn as sl

import sklearn.metrics as sm

import statsmodels.api as sm

import statsmodels.formula.api as smf

pip install statsmodels

#load the dataset

df = pd.read\_csv("medical clean 1.1.23.csv")

#Check total rows/columns

rows = df.shape[0]

cols = df.shape[1]

# display the number of rows and columns

print("Number of Rows: " + str(rows))

print("Number of Columns: " + str(cols))

#check for nan and null values

df.isna().any()

df.isnull().any()

# count of unique values in each column

print(df.nunique())

#drop all columns and rows not being used

to\_drop = ['CaseOrder', 'Customer\_id','Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Children', 'Age', 'Income', 'Marital', 'VitD\_levels', 'Soft\_drink', 'Initial\_admin', 'Complication\_risk', 'Full\_meals\_eaten', 'vitD\_supp', 'Allergic\_rhinitis', 'Services', 'Reflux\_esophagitis', 'Additional\_charges', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8', 'TotalCharge', 'Initial\_days']

df.drop(to\_drop, inplace=True, axis=1)

#check data types

df.dtypes

#Change object to category for ReAdmis

df["ReAdmis"] = df["ReAdmis"].astype('category')

#Change object to category for Anxiety

df["Anxiety"] = df["Anxiety"].astype('category')

#Change object to category for Overweight

df["Overweight"] = df["Overweight"].astype('category')

#Change object to category for Arthritis

df["Arthritis"] = df["Arthritis"].astype('category')

#Change object to category for HighBlood

df["HighBlood"] = df["HighBlood"].astype('category')

#Change object to category for Stroke

df["Stroke"] = df["Stroke"].astype('category')

#Change object to category for Gender

df["Gender"] = df["Gender"].astype('category')

#Change object to category for Diabetes

df["Diabetes"] = df["Diabetes"].astype('category')

#Change object to category for Asthma

df["Asthma"] = df["Asthma"].astype('category')

#Change object to category for back pain

df["BackPain"] = df["BackPain"].astype('category')

#Change object to category for Hyperlipidemia

df["Hyperlipidemia"] = df["Hyperlipidemia"].astype('category')

#Run get dummies on categorical

pd.get\_dummies(df, columns = ['Gender', 'ReAdmis', 'Anxiety', 'Overweight', 'HighBlood','Arthritis', 'Stroke', 'Diabetes', 'Asthma', 'Hyperlipidemia', 'BackPain'])

#create new df with the get dummies responses

dfupdated = pd.get\_dummies(df, columns = ['ReAdmis', 'Anxiety', 'Overweight', 'Arthritis','HighBlood', 'Gender', 'Stroke', 'Diabetes', 'Asthma', 'Hyperlipidemia', 'BackPain'])

#Drop columns to avoid dummy variable trap

to\_drop = ['Gender\_Male', 'Gender\_Nonbinary','ReAdmis\_No', 'Anxiety\_No', 'Overweight\_No', 'Stroke\_No', 'Diabetes\_No', 'Asthma\_No', 'Hyperlipidemia\_No', 'BackPain\_No', 'Arthritis\_No', 'HighBlood\_No']

dfupdated.drop(to\_drop, inplace=True, axis=1)

#Run a describe

dfupdated.describe()

# Import statistics Library

import statistics

#view the mean

print(dfupdated.mean())

print(dfupdated['ReAdmis\_Yes'].mode())

print(dfupdated['Anxiety\_Yes'].mode())

print(dfupdated['Overweight\_Yes'].mode())

print(dfupdated['Arthritis\_Yes'].mode())

print(dfupdated['HighBlood\_Yes'].mode())

print(dfupdated['Asthma\_Yes'].mode())

print(dfupdated['Gender\_Female'].mode())

print(dfupdated['Stroke\_Yes'].mode())

print(dfupdated['Diabetes\_Yes'].mode())

print(dfupdated['Hyperlipidemia\_Yes'].mode())

print(dfupdated['BackPain\_Yes'].mode())

print(dfupdated['Doc\_visits'].mode())

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

Doc\_visits = dfupdated["Doc\_visits"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,Doc\_visits)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

Gender\_Female = dfupdated["Gender\_Female"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,Gender\_Female)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

Anxiety\_Yes = dfupdated["Anxiety\_Yes"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,Anxiety\_Yes)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

Overweight\_Yes = dfupdated["Overweight\_Yes"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,Overweight\_Yes)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

Stroke\_Yes = dfupdated["Stroke\_Yes"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,Stroke\_Yes)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

Diabetes\_Yes = dfupdated["Diabetes\_Yes"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,Diabetes\_Yes)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

Asthma\_Yes = dfupdated["Asthma\_Yes"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,Asthma\_Yes)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

Hyperlipidemia\_Yes = dfupdated["Hyperlipidemia\_Yes"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,Hyperlipidemia\_Yes)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

BackPain\_Yes = dfupdated["BackPain\_Yes"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,BackPain\_Yes)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

HighBlood\_Yes = dfupdated["HighBlood\_Yes"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,HighBlood\_Yes)

spearmanr\_coff

#Check independence of each variable with Spearman Coefficient

ReAdmis\_Yes= dfupdated["ReAdmis\_Yes"]

Arthritis\_Yes = dfupdated["Arthritis\_Yes"]

#Find the Spearmen Cofficient.

spearmanr\_coff, p\_value = spearmanr(ReAdmis\_Yes,Arthritis\_Yes)

spearmanr\_coff

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='Doc\_visits', data=dfupdated)

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='Gender\_Female', data=dfupdated)

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='Anxiety\_Yes', data=dfupdated)

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='HighBlood\_Yes', data=dfupdated)

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='Hyperlipidemia\_Yes', data=dfupdated)

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='Stroke\_Yes', data=dfupdated)

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='Overweight\_Yes', data=dfupdated)

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='Diabetes\_Yes', data=dfupdated)

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='BackPain\_Yes', data=dfupdated)

# Check the Prevalence of Each Classification Category with visual plot

sns.countplot(x='Asthma\_Yes', data=dfupdated)

# Predictor Variable Plot

sns.countplot(x='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions to patients who have diabetes

sns.countplot(x='Diabetes\_Yes', hue='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions to patients who have had a stroke

sns.countplot(x='Stroke\_Yes', hue='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions to patients who have hyperlipidemia

sns.countplot(x='Hyperlipidemia\_Yes', hue='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions to doctor visits

sns.countplot(x='Doc\_visits', hue='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions patients with anxiety

sns.countplot(x='Anxiety\_Yes', hue='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions patients with asthma

sns.countplot(x='Asthma\_Yes', hue='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions patients with back pain

sns.countplot(x='BackPain\_Yes', hue='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions for female patients

sns.countplot(x='Gender\_Female', hue='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions for overweight patients

sns.countplot(x='Overweight\_Yes', hue='ReAdmis\_Yes', data=dfupdated)

#Compare readmissions for patients with high blood pressure

sns.countplot(x='HighBlood\_Yes', hue='ReAdmis\_Yes', data=dfupdated)

# plotting histogram

plt.hist(dfupdated['ReAdmis\_Yes'],bins = 35,

alpha = 0.45, color = 'red')

plt.title("ReAdmis\_Yes")

plt.show()

# plotting histogram

plt.hist(dfupdated[“Doc\_visits”],bins = 35,

alpha = 0.45, color = 'red')

plt.title("Doc\_visits")

plt.show()

# plotting histogram

plt.hist(dfupdated['Stroke\_Yes'],bins = 35,

alpha = 0.45, color = 'red')

plt.title("Stroke\_Yes")

plt.show()

# plotting histogram

plt.hist(dfupdated['Anxiety\_Yes'],bins = 35,

alpha = 0.45, color = 'red')

plt.title("Anxiety\_Yes")

plt.show()

# plotting histogram

plt.hist(dfupdated[‘HighBlood\_Yes’],bins = 35,

alpha = 0.45, color = 'red')

plt.title("HighBlood\_Yes")

plt.show()

# plotting histogram

plt.hist(dfupdated[‘Diabetes\_Yes’],bins = 35,

alpha = 0.45, color = 'red')

plt.title("Diabetes\_Yes")

plt.show()

# plotting histogram

plt.hist(dfupdated[‘Hyperlipidemia\_Yes’],bins = 35,

alpha = 0.45, color = 'red')

plt.title("Hyperlipidemia\_Yes")

plt.show()

# plotting histogram

plt.hist(dfupdated[‘Arthritis\_Yes’],bins = 35,

alpha = 0.45, color = 'red')

plt.title("Arthritis\_Yes")

plt.show()

# plotting histogram

plt.hist(dfupdated[‘Overweight\_Yes’],bins = 35,

alpha = 0.45, color = 'red')

plt.title("Overweight\_Yes")

plt.show()

# plotting histogram

plt.hist(dfupdated[‘Gender\_Female’],bins = 35,

alpha = 0.45, color = 'red')

plt.title("Gender\_Female")

plt.show()

# plotting histogram

plt.hist(dfupdated[‘BackPain\_Yes’],bins = 35,

alpha = 0.45, color = 'red')

plt.title("BackPain\_Yes")

plt.show()

# plotting histogram

plt.hist(dfupdated[‘Asthma\_Yes’],bins = 35,

alpha = 0.45, color = 'red')

plt.title("Asthma\_Yes")

plt.show()

#boxplots

df = pd.DataFrame(data = np.random.random(size=(4,4)), columns = ['Doc\_visits', 'Stroke\_Yes', 'ReAdmis\_Yes', 'Anxiety\_Yes'])

df.plot(kind='box')

plt.show()

df = pd.DataFrame(data = np.random.random(size=(4,4)), columns = ['HighBlood\_Yes', 'Diabetes\_Yes', 'Hyperlipidemia\_Yes', 'Arthritis\_Yes'])

df.plot(kind='box')

plt.show()

df = pd.DataFrame(data = np.random.random(size=(4,3)), columns = ['Overweight\_Yes', 'Gender\_Female', 'BackPain\_Yes'])

df.plot(kind='box')

plt.show()

df = pd.DataFrame(data = np.random.random(size=(4,2)), columns = ['Asthma\_Yes','ReAdmis\_Yes'])

df.plot(kind='box')

plt.show()

#how does the Y data stack up?

dfupdated['ReAdmis\_Yes'].value\_counts()

sns.countplot(x='ReAdmis\_Yes', data=dfupdated, palette='hls')

plt.show()

plt.savefig('count\_plot')

#review percentage of patients who readmitted/versus not readmitted

count\_no\_sub = len(dfupdated[dfupdated['ReAdmis\_Yes']==0])

count\_sub = len(dfupdated[dfupdated['ReAdmis\_Yes']==1])

pct\_of\_no\_sub = count\_no\_sub/(count\_no\_sub+count\_sub)

print("patients who did not readmit", pct\_of\_no\_sub\*100)

pct\_of\_sub = count\_sub/(count\_no\_sub+count\_sub)

print("patients who readmitted", pct\_of\_sub\*100)

#Check the means

dfupdated.groupby('ReAdmis\_Yes').mean()

#review histogram visual aides

dfupdated.Anxiety\_Yes.hist()

plt.title('Histogram of ReAdmittance with Anxiety')

plt.xlabel('Anxiety\_Yes')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_anxiety')

dfupdated.Doc\_visits.hist()

plt.title('Histogram of ReAdmittance with doc visits')

plt.xlabel('Doc\_visits')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_docvisits')

dfupdated.Gender\_Female.hist()

plt.title('Histogram of ReAdmittance for gender\_female')

plt.xlabel('Gender\_Female')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_GenderFemale')

dfupdated.Diabetes\_Yes.hist()

plt.title('Histogram of ReAdmittance for Diabetes\_Yes')

plt.xlabel('Diabetes\_Yes')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_DiabetesYes')

dfupdated.Stroke\_Yes.hist()

plt.title('Histogram of ReAdmittance for Stroke\_Yes')

plt.xlabel('Stroke\_Yes')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_StrokeYes')

dfupdated.Overweight\_Yes.hist()

plt.title('Histogram of ReAdmittance for Overweight\_Yes')

plt.xlabel('Overweight\_Yes')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_OverweightYes')

dfupdated.Hyperlipidemia\_Yes.hist()

plt.title('Histogram of ReAdmittance for Hyperlipidemia\_Yes')

plt.xlabel('Hyperlipidemia\_Yes')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_HyperlipidemiaYes')

dfupdated.Asthma\_Yes.hist()

plt.title('Histogram of ReAdmittance for Asthma\_Yes')

plt.xlabel('Asthma\_Yes')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_AsthmaYes')

dfupdated.HighBlood\_Yes.hist()

plt.title('Histogram of ReAdmittance for HighBlood\_Yes')

plt.xlabel('HighBlood\_Yes')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_HighBloodYes')

dfupdated.BackPain\_Yes.hist()

plt.title('Histogram of ReAdmittance for BackPain\_Yes')

plt.xlabel('BackPain\_Yes')

plt.ylabel('ReAdmis\_Yes')

plt.savefig('hist\_BackPainYes')

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Overweight\_Yes + Stroke\_Yes + HighBlood\_Yes + Anxiety\_Yes + Diabetes\_Yes + Hyperlipidemia\_Yes + Arthritis\_Yes + Gender\_Female + BackPain\_Yes + Asthma\_Yes + Doc\_visits", data=dfupdated).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#define the predictor variables and the response variable

X = dfupdated[['Overweight\_Yes', 'Stroke\_Yes', 'HighBlood\_Yes', 'Anxiety\_Yes', 'Diabetes\_Yes', 'Hyperlipidemia\_Yes', 'Arthritis\_Yes', 'Gender\_Female', 'BackPain\_Yes', 'Asthma\_Yes', 'Doc\_visits']]

y = dfupdated['ReAdmis\_Yes']

#split the dataset into training (70%) and testing (30%) sets

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

#instantiate the model

log\_regression = LogisticRegression()

#fit the model using the training data

log\_regression.fit(X\_train,y\_train)

#define metrics

y\_pred\_proba = log\_regression.predict\_proba(X\_test)[::,1]

fpr, tpr, \_ = metrics.roc\_curve(y\_test, y\_pred\_proba)

auc = metrics.roc\_auc\_score(y\_test, y\_pred\_proba)

#create ROC curve

plt.plot(fpr,tpr,label="AUC="+str(auc))

plt.legend(loc=4)

plt.show()

#plot confusion matrix

actual = np.random.binomial(1,.9,size = 1000)

predicted = np.random.binomial(1,.9,size = 1000)

confusion\_matrix = metrics.confusion\_matrix(actual, predicted)

cm\_display = metrics.ConfusionMatrixDisplay(confusion\_matrix = confusion\_matrix, display\_labels = [False, True])

cm\_display.plot()

plt.show()

#drop highest p-value

to\_drop = ['Doc\_visits']

dfupdated.drop(to\_drop, inplace=True, axis=1)

#new dataframe for second model

dfupdated2 = dfupdated[['ReAdmis\_Yes','Stroke\_Yes', 'HighBlood\_Yes', 'Anxiety\_Yes', 'Hyperlipidemia\_Yes', 'Arthritis\_Yes', 'Overweight\_Yes', 'Gender\_Female', 'BackPain\_Yes', 'Asthma\_Yes', 'Diabetes\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Overweight\_Yes + Stroke\_Yes + HighBlood\_Yes + Anxiety\_Yes + Diabetes\_Yes + Hyperlipidemia\_Yes + Arthritis\_Yes + Gender\_Female + BackPain\_Yes + Asthma\_Yes", data=dfupdated2).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#new dataframe for third model

dfupdated3 = dfupdated[['ReAdmis\_Yes', 'HighBlood\_Yes', 'Anxiety\_Yes', 'Hyperlipidemia\_Yes', 'Arthritis\_Yes', 'Overweight\_Yes', 'Gender\_Female', 'BackPain\_Yes', 'Asthma\_Yes', 'Diabetes\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Overweight\_Yes + HighBlood\_Yes + Anxiety\_Yes + Diabetes\_Yes + Hyperlipidemia\_Yes + Arthritis\_Yes + Gender\_Female + BackPain\_Yes + Asthma\_Yes", data=dfupdated2).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#new dataframe for 4th model

Dfupdated4 = dfupdated[['ReAdmis\_Yes', 'HighBlood\_Yes', 'Hyperlipidemia\_Yes', 'Arthritis\_Yes', 'Overweight\_Yes', 'Gender\_Female', 'BackPain\_Yes', 'Asthma\_Yes', 'Diabetes\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Overweight\_Yes + HighBlood\_Yes + Diabetes\_Yes + Hyperlipidemia\_Yes + Arthritis\_Yes + Gender\_Female + BackPain\_Yes + Asthma\_Yes", data=Dfupdated4).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#new dataframe for 5th model

Dfupdated5 = dfupdated[['ReAdmis\_Yes', 'Hyperlipidemia\_Yes', 'Arthritis\_Yes', 'Overweight\_Yes', 'Gender\_Female', 'BackPain\_Yes', 'Asthma\_Yes', 'Diabetes\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Overweight\_Yes + Diabetes\_Yes + Hyperlipidemia\_Yes + Arthritis\_Yes + Gender\_Female + BackPain\_Yes + Asthma\_Yes", data=Dfupdated5).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#new dataframe for 6th model

Dfupdated6 = dfupdated[['ReAdmis\_Yes', 'Hyperlipidemia\_Yes', 'Arthritis\_Yes', 'Overweight\_Yes', 'Gender\_Female', 'BackPain\_Yes', 'Asthma\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Overweight\_Yes + Hyperlipidemia\_Yes + Arthritis\_Yes + Gender\_Female + BackPain\_Yes + Asthma\_Yes", data=Dfupdated6).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#new dataframe for 7th model

Dfupdated7 = dfupdated[['ReAdmis\_Yes', 'Arthritis\_Yes', 'Overweight\_Yes', 'Gender\_Female', 'BackPain\_Yes', 'Asthma\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Overweight\_Yes + Arthritis\_Yes + Gender\_Female + BackPain\_Yes + Asthma\_Yes", data=Dfupdated7).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#new dataframe for 8th model

Dfupdated8 = dfupdated[['ReAdmis\_Yes', 'Overweight\_Yes', 'Gender\_Female', 'BackPain\_Yes', 'Asthma\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Overweight\_Yes + Gender\_Female + BackPain\_Yes + Asthma\_Yes", data=Dfupdated8).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#new dataframe for 9th model

Dfupdated9 = dfupdated[['ReAdmis\_Yes', 'Gender\_Female', 'BackPain\_Yes', 'Asthma\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Gender\_Female + BackPain\_Yes + Asthma\_Yes", data=Dfupdated9).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#new dataframe for 10th model

Dfupdated10 = dfupdated[['ReAdmis\_Yes', 'BackPain\_Yes', 'Asthma\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ BackPain\_Yes + Asthma\_Yes", data=Dfupdated10).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#new dataframe for final model

DfupdatedFinal = dfupdated[['ReAdmis\_Yes', 'Asthma\_Yes']]

#Prepare and fit the model

log\_reg = smf.logit("ReAdmis\_Yes ~ Asthma\_Yes", data=DfupdatedFinal).fit()

#print the model

# Summary of results

print(log\_reg.summary())

#Run confusion matrix on final model

actual = np.random.binomial(1,.9,size = 1000)

predicted = np.random.binomial(1,.9,size = 1000)

confusion\_matrix = metrics.confusion\_matrix(actual, predicted)

cm\_display = metrics.ConfusionMatrixDisplay(confusion\_matrix = confusion\_matrix, display\_labels = [False, True])

cm\_display.plot()

plt.show()