In [1]:
```
# Import packages that will be used for the logistics regression analysis
import pylab
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 statsmodels.graphics.mosaicplot import mosaic
from statsmodels.stats.outliers_influence import variance_inflation_factor
from IPython.core.display import HTML
from IPython.display import display
import pandas as pd
from pandas.api.types import CategoricalDtype
from pandas import Series, DataFrame
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE

# Import data set that will be used for the logistics regression analysis
pd.set_option('display.max_columns', None)
df = pd.read_csv (r'C:\Users\fahim\Documents\0_WGUDocuments\d208\1medical_clean.csv')
# Check data types and number of values, as well as overall size of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CaseOrder           10000 non-null  int64  
 1   Customer_id         10000 non-null  object 
 2   Interaction         10000 non-null  object 
 3   UID                 10000 non-null  object 
 4   City                10000 non-null  object 
 5   State               10000 non-null  object 
 6   County              10000 non-null  object 
 7   Zip                 10000 non-null  int64  
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64  
 11  Area                10000 non-null  object 
 12  TimeZone            10000 non-null  object 
 13  Job                 10000 non-null  object 
 14  Children            10000 non-null  int64  
 15  Age                 10000 non-null  int64  
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object 
 18  Gender              10000 non-null  object 
 19  ReAdmis             10000 non-null  object 
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64  
 22  Full_meals_eaten    10000 non-null  int64  
 23  vitD_supp           10000 non-null  int64  
 24  Soft_drink          10000 non-null  object 
 25  Initial_admin       10000 non-null  object 
 26  HighBlood           10000 non-null  object 
 27  Stroke              10000 non-null  object 
 28  Complication_risk   10000 non-null  object 
 29  Overweight          10000 non-null  object 
 30  Arthritis           10000 non-null  object 
 31  Diabetes            10000 non-null  object 
 32  Hyperlipidemia      10000 non-null  object 
 33  BackPain            10000 non-null  object 
 34  Anxiety             10000 non-null  object 
 35  Allergic_rhinitis   10000 non-null  object 
 36  Reflux_esophagitis  10000 non-null  object 
 37  Asthma              10000 non-null  object 
 38  Services            10000 non-null  object 
 39  Initial_days        10000 non-null  float64
```

```
 40  TotalCharge          10000 non-null  float64
 41  Additional_charges   10000 non-null  float64
 42  Item1                10000 non-null  int64
 43  Item2                10000 non-null  int64
 44  Item3                10000 non-null  int64
 45  Item4                10000 non-null  int64
 46  Item5                10000 non-null  int64
 47  Item6                10000 non-null  int64
 48  Item7                10000 non-null  int64
 49  Item8                10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [2]:
```
# Visually inspect dataframe to facilitate exploration, spot problems
pd.set_option("display.max_columns", None)
df.head(5)
```

Out[2]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Popul
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL	Morgan	35621	34.34960	-86.72508	
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL	Jackson	32446	30.84513	-85.22907	
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD	Minnehaha	57110	43.54321	-96.63772	
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN	Waseca	56072	43.89744	-93.51479	
4	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA	King William	23181	37.59894	-76.88958	

In [3]:
```
#check if there is any duplicate data entries present in columns
df[df.duplicated()]
```

Out[3]:

CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Population	Area	TimeZone	Job	Children	Age	Income	I

In [4]:
```
# check if there are any duplicated columns in the data set - if there are none then the output should be False
df.columns.duplicated().any()
```

Out[4]:
```
False
```

In [5]:
```
# check if there are any duplicated rows in the data set - if there are none then the output should be False
df.duplicated().any()
```

Out[5]:
```
False
```

In [6]:
```
#Summary Statistics
df.Age.describe()
```

Out[6]:
```
count    10000.000000
mean        53.511700
std         20.638538
min         18.000000
25%         36.000000
50%         53.000000
75%         71.000000
max         89.000000
Name: Age, dtype: float64
```

In [7]:
```
df.Gender.value_counts()
```

Out[7]:
```
Female       5018
Male         4768
Nonbinary     214
Name: Gender, dtype: int64
```

In [8]:
```
df.VitD_levels.describe()
```

Out[8]:
```
count    10000.000000
mean        17.964262
std          2.017231
min          9.806483
25%         16.626439
50%         17.951122
75%         19.347963
max         26.394449
Name: VitD_levels, dtype: float64
```

In [9]:
```
df.Initial_admin.value_counts().sort_index()
```

Out[9]:
```
Elective Admission       2504
Emergency Admission      5060
Observation Admission    2436
Name: Initial_admin, dtype: int64
```

In [10]:
```
df.HighBlood.value_counts()
```

Out[10]:
```
No     5910
Yes    4090
Name: HighBlood, dtype: int64
```

In [11]:
```
df.Complication_risk.value_counts().sort_index()
```

Out[11]:
```
High      3358
Low       2125
Medium    4517
Name: Complication_risk, dtype: int64
```

In [12]:
```
df.Overweight.value_counts()
```

Out[12]:
```
Yes    7094
No     2906
Name: Overweight, dtype: int64
```

In [13]:
```
df.BackPain.value_counts()
```

Out[13]:
```
No     5886
Yes    4114
Name: BackPain, dtype: int64
```

In [14]:
```
df.Diabetes.value_counts()
```

Out[14]:
```
No     7262
Yes    2738
Name: Diabetes, dtype: int64
```

In [15]:
```
df.Asthma.value_counts()
```

Out[15]:
```
No     7107
Yes    2893
Name: Asthma, dtype: int64
```

In [16]:
```
df.Initial_days.describe()
```

Out[16]:
```
count    10000.000000
mean        34.455299
std         26.309341
min          1.001981
25%          7.896215
50%         35.836244
75%         61.161020
max         71.981490
Name: Initial_days, dtype: float64
```

In [17]:
```
df.Initial_days.nlargest(n=20)
```

Out[17]:
```
7968    71.98149
5326    71.96869
7479    71.96546
6166    71.96415
8066    71.96342
5874    71.96164
5829    71.96134
9159    71.95813
8817    71.95472
7524    71.94732
9074    71.94459
7839    71.92930
9677    71.92647
9221    71.92413
5162    71.92171
9101    71.90712
9766    71.90694
5374    71.90056
6601    71.89863
7214    71.89805
Name: Initial_days, dtype: float64
```

In [18]:
```
df.Arthritis.value_counts()
```

Out[18]:
```
No     6426
Yes    3574
Name: Arthritis, dtype: int64
```

In [19]:
```
# Data Preparation for analysis
# Convert column to category from string
df["TimeZone"] = df["TimeZone"].astype("category")
# Reformat column representing currency in USD to 3 decimal places from 6
df["Income"] = df["Income"].astype(int)
# Convert column to category from string
df["Marital"] = df["Marital"].astype("category")
# Convert column to category from string
df["Gender"] = df["Gender"].astype("category")

# Convert categorical yes/no values to numeric 1/0 values
df = df.replace(to_replace = ['Yes','No'],value = [1,0])

# Perform one-hot encoding
# Generate columns of dummy values for dataframe's Gender column
gender_temp_df = pd.get_dummies(data=df["Gender"], drop_first=True)
# Generate columns of dummy values for dataframe's Initial_admin column
initial_admit_temp_df = pd.get_dummies(data=df["Initial_admin"], drop_first=True)
# Generate columns of dummy values for dataframe's Complication_risk column
comp_risk_temp_df = pd.get_dummies(data=df["Complication_risk"], drop_first=True)
# Create the new df with the variables used for this analysis
regress_df = df[["Age", "VitD_levels", "HighBlood", "Overweight", "Arthritis", "Diabetes", "BackPain", "Asthma", "Initi
# Generate and apply new Pythonic names for ease of use
pythonic_columns = ["age", "vit_d_level", "high_bp", "overweight", "arthritis", "diabetes", "back_pain", "asthma", "day
regress_df.set_axis(pythonic_columns, axis=1, inplace=True)
# Insert the generated dummy variables to new dataframe, placing them in the same order as the original dataframe
# Dummies for Complication Risk
regress_df.insert(4, "comp_risk_medium", comp_risk_temp_df.Medium)
regress_df.insert(4, "comp_risk_low", comp_risk_temp_df.Low)
# Dummies for Initial Admit
regress_df.insert(3, "initial_admit_emerg", initial_admit_temp_df["Emergency Admission"])
regress_df.insert(3, "initial_admit_observ", initial_admit_temp_df["Observation Admission"])
# Dummies for Gender
regress_df.insert(2, "gender_nonbinary", gender_temp_df.Nonbinary)
regress_df.insert(2, "gender_male", gender_temp_df.Male)
# Check resulting dataframe
regress_df
```

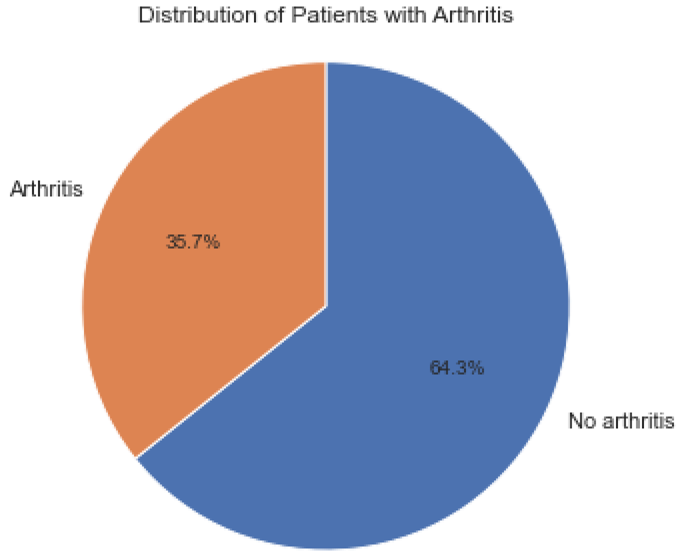
Out[19]:

	age	vit_d_level	gender_male	gender_nonbinary	high_bp	initial_admit_observ	initial_admit_emerg	overweight	comp_risk_low	comp_
0	53	19.141466	1	0	1	0	1	0	0	
1	51	18.940352	0	0	1	0	1	1	0	
2	53	18.057507	0	0	1	0	0	1	0	
3	78	16.576858	1	0	0	0	0	0	0	
4	22	17.439069	0	0	0	0	0	0	1	
...	...	...	...	...	...	...	...	...	...	
9995	25	16.980860	1	0	1	0	1	0	0	
9996	87	18.177020	1	0	1	0	0	1	0	
9997	45	17.129070	0	0	1	0	0	1	0	
9998	43	19.910430	1	0	0	0	1	1	0	
9999	70	18.388620	0	0	0	1	0	1	1	

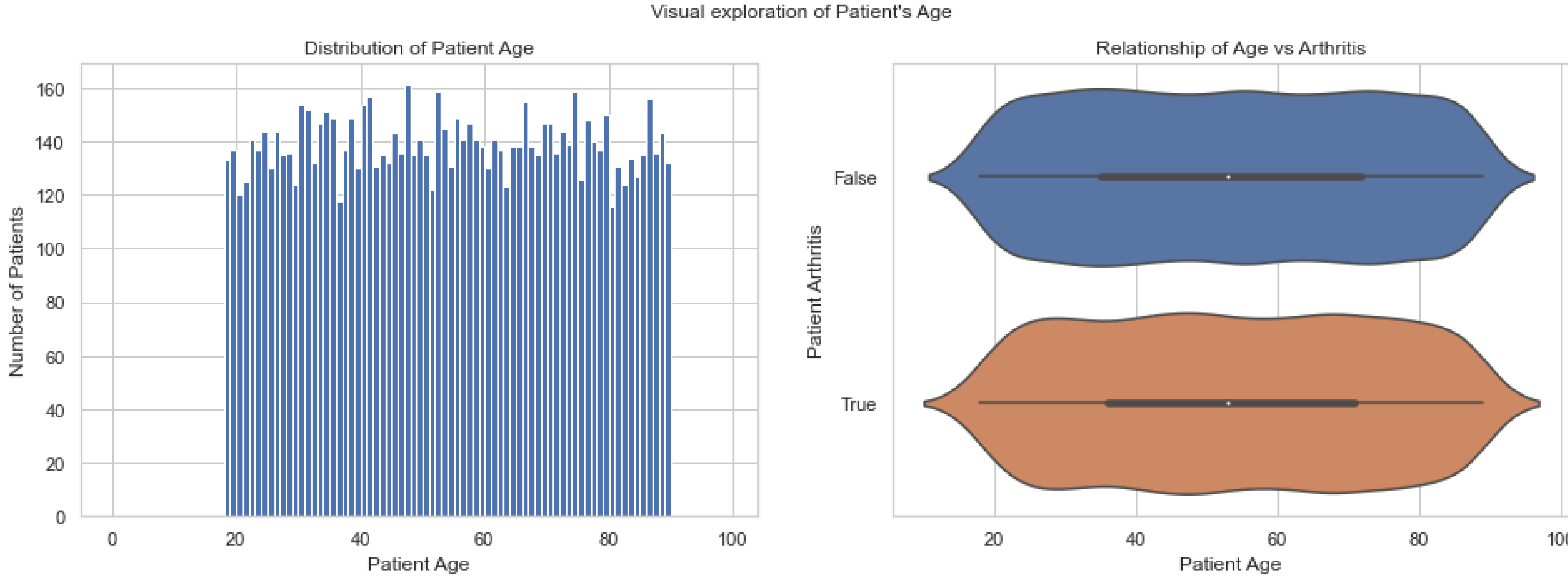
10000 rows × 15 columns

In [20]:

```
#Bivariate distribution of Arthritis
plt.figure(figsize = [16,5])
plt.title('Distribution of Patients with Arthritis')
arthritis_counts = regress_df.arthritis.value_counts()
arthritis_labels = ["No arthritis", "Arthritis"]
plt.pie(arthritis_counts, labels=arthritis_labels, autopct='%1.1f%%', startangle=90, counterclock=False)
plt.axis('square');
```

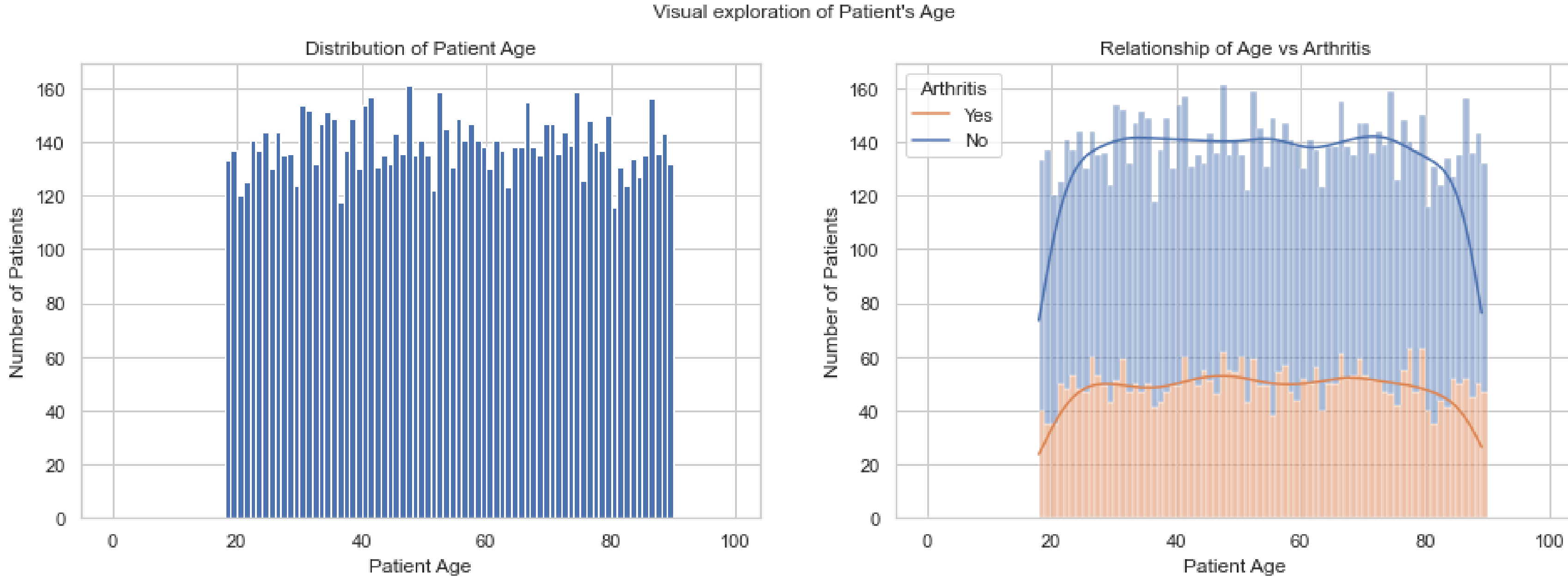


In [21]:
```
#Univariate and bivariate distribution of Age
plt.figure(figsize = [16,5])
plt.suptitle("Visual exploration of Patient's Age")

# LEFT plot: Univariate exploration of age
plt.subplot(1, 2, 1)
plt.title('Distribution of Patient Age')
bins = np.arange(0, 100, 1)
plt.hist(data=regress_df, x="age", bins=bins)
plt.xlabel('Patient Age')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of age vs arthritis
plt.subplot(1, 2, 2)
plt.title("Relationship of Age vs Arthritis")
sb.violinplot(data = regress_df, x="age", y="arthritis", orient='h')
plt.xlabel("Patient Age")
plt.ylabel("Patient Arthritis")
plt.yticks([0,1], ["False", "True"]);
```

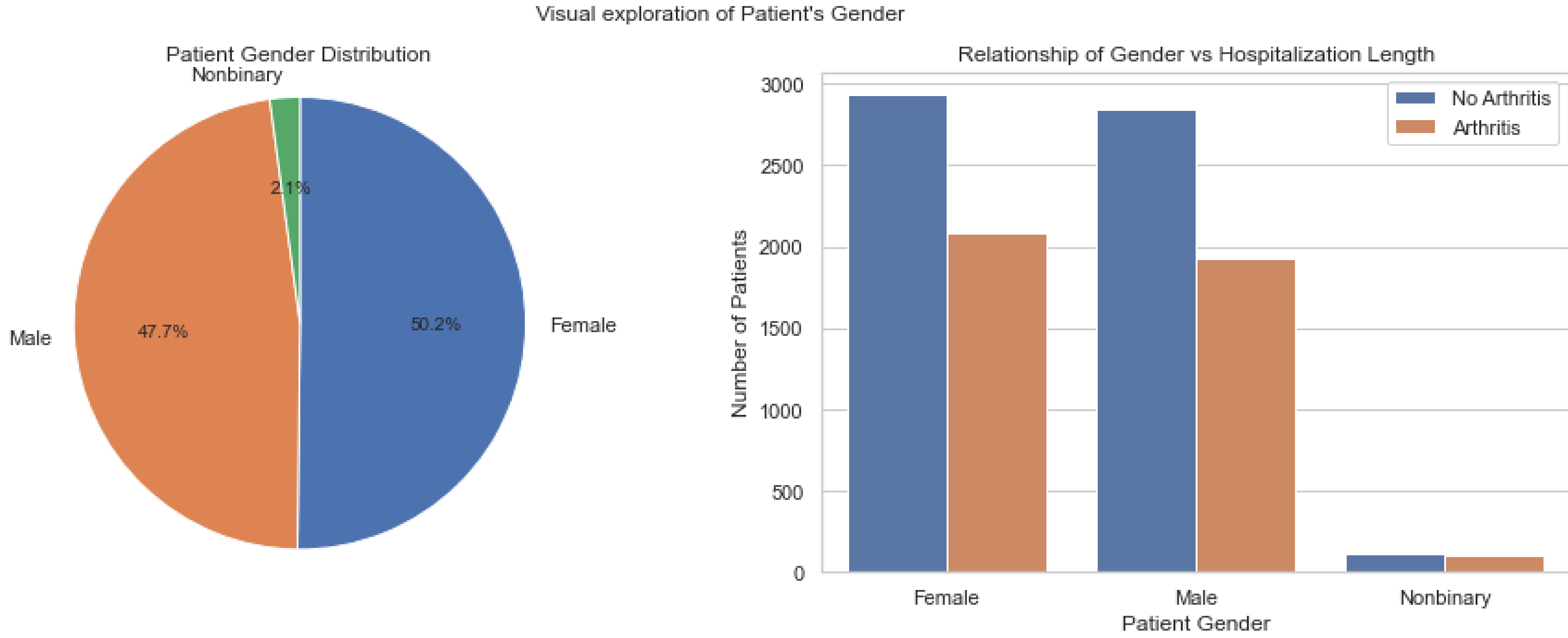


In [22]:

```
plt.figure(figsize = [16,5])
plt.suptitle("Visual exploration of Patient's Age")

# LEFT plot: Univariate exploration of age
plt.subplot(1, 2, 1)
plt.title('Distribution of Patient Age')
bins = np.arange(0, 100, 1)
plt.hist(data=regress_df, x="age", bins=bins)
plt.xlabel('Patient Age')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of age vs arthritis
plt.subplot(1, 2, 2)
plt.title("Relationship of Age vs Arthritis")
sb.histplot(data = regress_df, x="age", hue="arthritis", bins=bins, kde=True, multiple="stack")
plt.legend(title="Arthritis", labels=["Yes", "No"])
plt.xlabel("Patient Age")
plt.ylabel("Number of Patients");
```

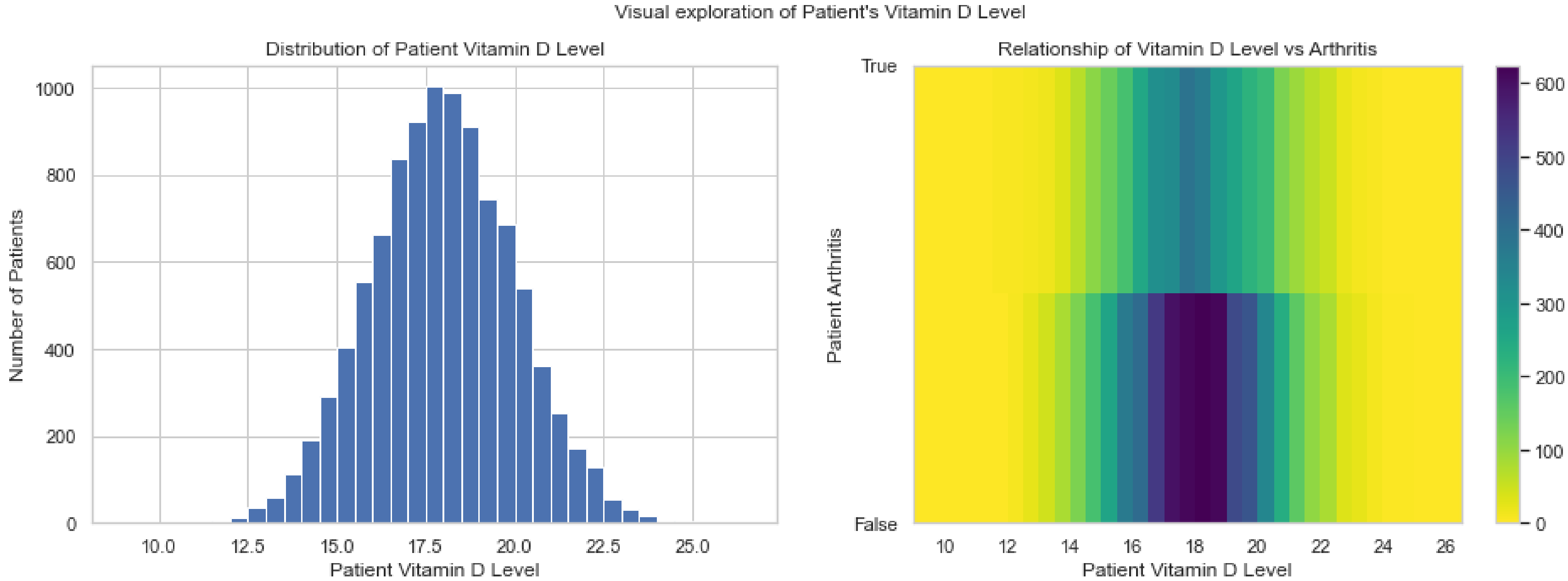


In [23]:
```
#Univariate and bivariate distribution of Gender
plt.figure(figsize = [16,5])
plt.suptitle("Visual exploration of Patient's Gender")

# LEFT plot: Univariate exploration of num_children
plt.subplot(1, 2, 1)
plt.title("Patient Gender Distribution")
gender_counts = df["Gender"].value_counts()
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# RIGHT plot: Bivariate exploration of num_children vs arthritis
plt.subplot(1, 2, 2)
plt.title("Relationship of Gender vs Hospitalization Length")
sb.countplot(data = df, x="Gender", hue="BackPain")
plt.legend(["No Arthritis", "Arthritis"])
plt.xlabel("Patient Gender")
plt.ylabel("Number of Patients");
```

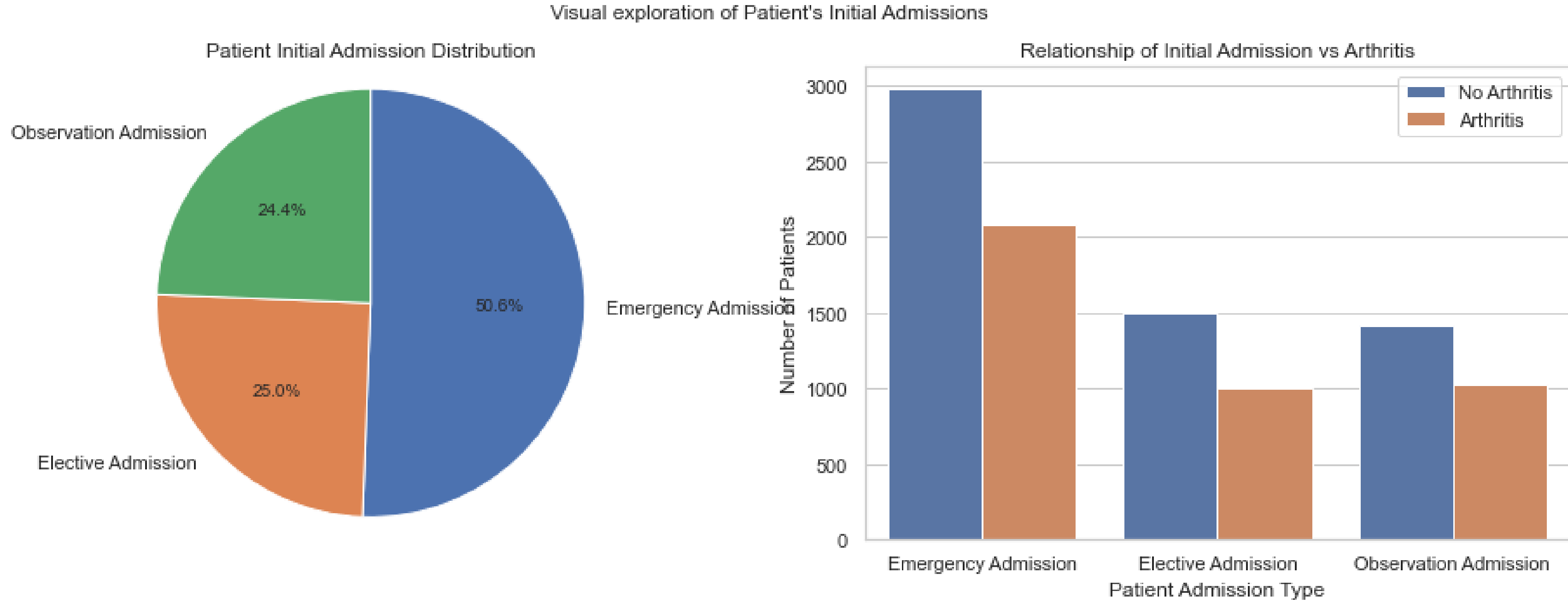


In [24]:

```
#Univariate and bivariate distribution of Vitamin D Level
plt.figure(figsize = [16,5])
plt.suptitle("Visual exploration of Patient's Vitamin D Level")

# LEFT plot: Univariate exploration of vit_d_level
plt.subplot(1, 2, 1)
plt.title('Distribution of Patient Vitamin D Level')
bins = np.arange(9, 27, 0.5)
plt.hist(data=regress_df, x="vit_d_level", bins=bins)
plt.xlabel('Patient Vitamin D Level')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of vit_d_level vs arthritis
plt.subplot(1, 2, 2)
plt.title("Relationship of Vitamin D Level vs Arthritis")
bins_y = np.arange(0, 1.25, 0.5)
plt.hist2d(data= regress_df, x="vit_d_level", y="arthritis", bins=[bins, bins_y], cmap= "viridis_r")
plt.colorbar()
plt.xlabel("Patient Vitamin D Level")
plt.ylabel("Patient Arthritis")
plt.yticks([0,1], ["False", "True"]);
```

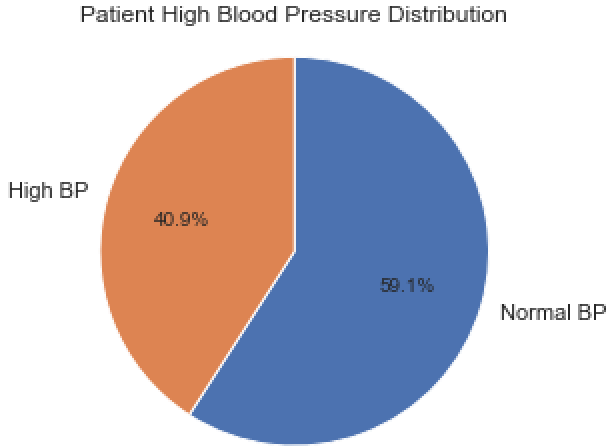


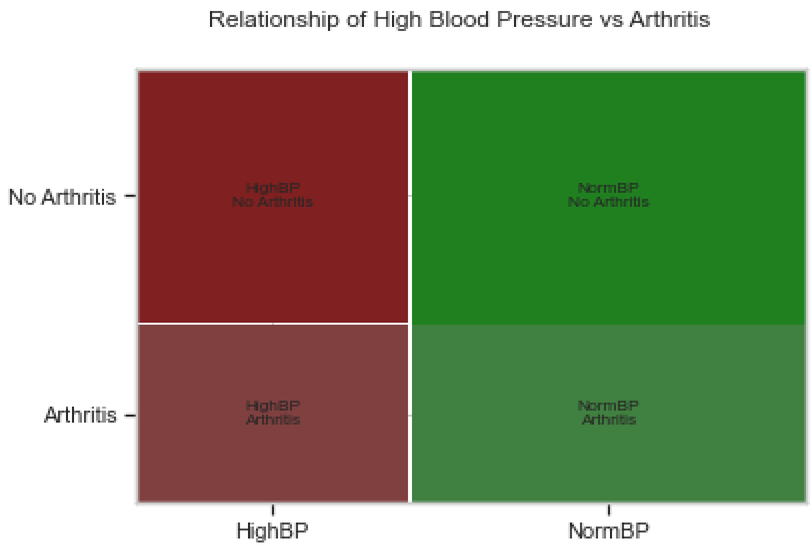
In [25]:
```
#Univariate and bivariate distribution of Initial Admissions
plt.figure(figsize = [16,5])
plt.suptitle("Visual exploration of Patient's Initial Admissions")

# LEFT plot: Univariate exploration of initial_admin
plt.subplot(1, 2, 1)
plt.title("Patient Initial Admission Distribution")
init_admit_counts = df["Initial_admin"].value_counts()
plt.pie(init_admit_counts, labels=init_admit_counts.index, autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# RIGHT plot: Bivariate exploration of Initial_admin vs arthritis
plt.subplot(1, 2, 2)
plt.title("Relationship of Initial Admission vs Arthritis")
sb.countplot(data = df, x="Initial_admin", hue="BackPain")
plt.legend(["No Arthritis", "Arthritis"])
plt.xlabel("Patient Admission Type")
plt.ylabel("Number of Patients");
```



In [26]:

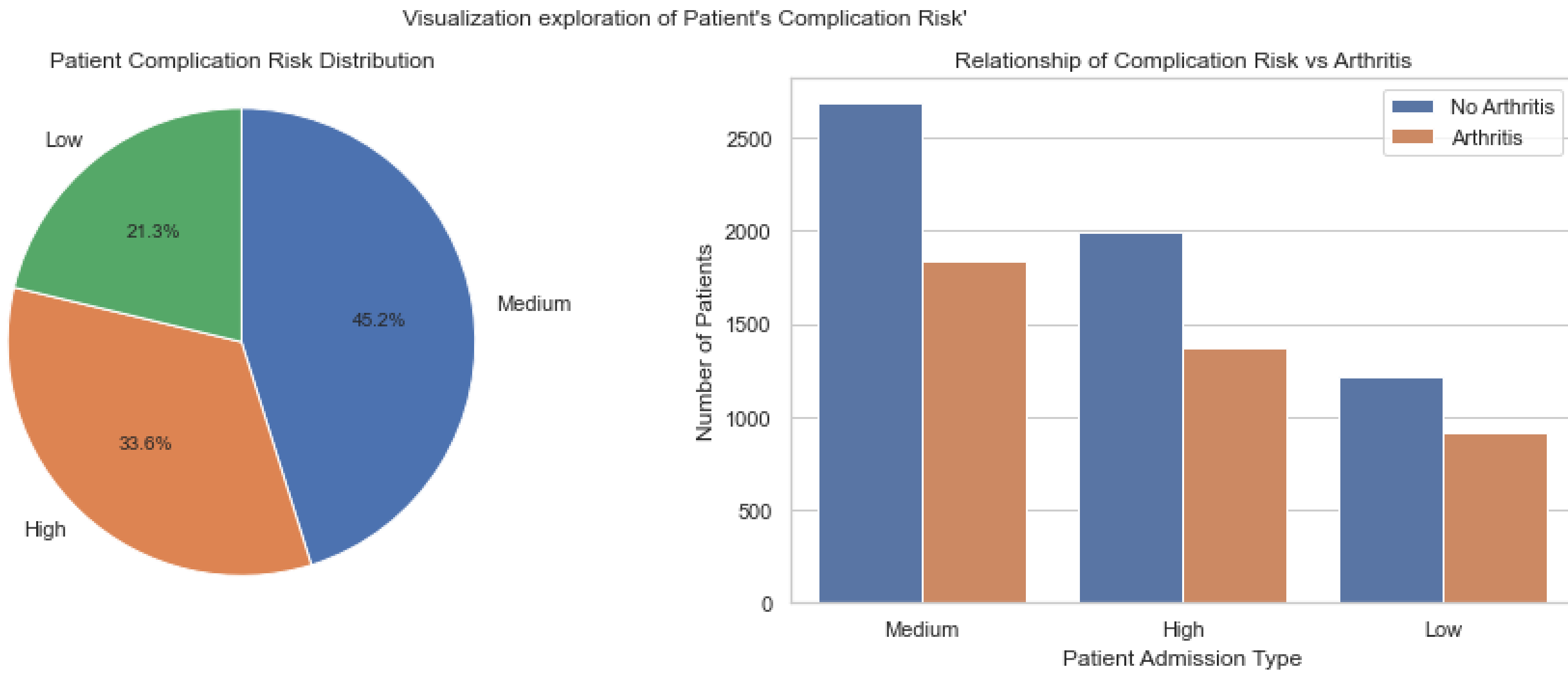
```
#Univariate and bivariate distribution of High Blood pressure
# TOP plot: Univariate exploration of high bp
plt.title("Patient High Blood Pressure Distribution")
high_bp_counts = df["HighBlood"].value_counts()
plt.pie(high_bp_counts, labels=["Normal BP", "High BP"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# BOTTOM plot: Bivariate exploration of high bp vs arthritis
temp_df = df[["HighBlood", "BackPain"]].copy()
high_bp_map = {1 : "HighBP", 0: "NormBP"}
arthritis_map = {1 : "Arthritis", 0: "No Arthritis"}
temp_df["HighBlood"] = temp_df["HighBlood"].map(high_bp_map)
temp_df["BackPain"] = temp_df["BackPain"].map(arthritis_map)
mosaic(temp_df, ["HighBlood", "BackPain"])
plt.suptitle("Relationship of High Blood Pressure vs Arthritis");
```





In [27]:
```
#Univariate and bivariate distribution of Complication Risk
plt.figure(figsize = [16,5])
plt.suptitle("Visualization exploration of Patient's Complication Risk'")

# LEFT plot: Univariate exploration of complication_risk
plt.subplot(1, 2, 1)
plt.title("Patient Complication Risk Distribution")
comp_risk_counts = df["Complication_risk"].value_counts()
comp_risk_labels = ["Medium", "High", "Low"]
plt.pie(comp_risk_counts, labels=comp_risk_counts.index, autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');
```

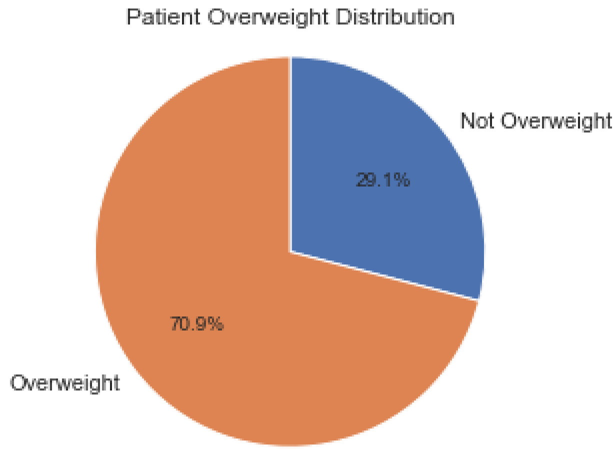
```
# RIGHT plot: Bivariate exploration of complication_risk vs arthritis
plt.subplot(1, 2, 2)
plt.title("Relationship of Complication Risk vs Arthritis")
sb.countplot(data = df, x="Complication_risk", hue="BackPain")
plt.legend(["No Arthritis", "Arthritis"])
plt.xlabel("Patient Admission Type")
plt.ylabel("Number of Patients");
```

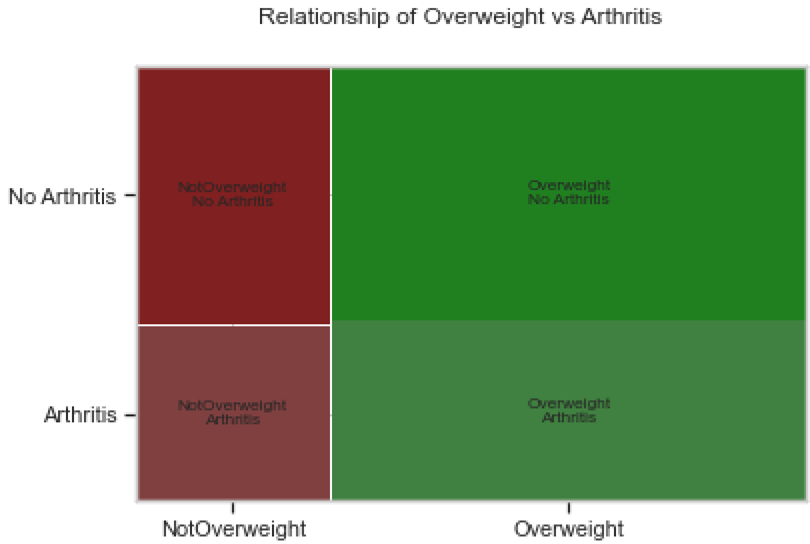


In [28]:

```
#Univariate and bivariate distribution of Overweight
# TOP plot: Univariate exploration of overweight
plt.title("Patient Overweight Distribution")
overweight_counts = df["Overweight"].value_counts().sort_index()
plt.pie(overweight_counts, labels=["Not Overweight", "Overweight"], autopct='%1.1f%%', startangle=90, counterclock = Fa
plt.axis('square');

# BOTTOM plot: Bivariate exploration of overweight vs arthritis
temp_df = df[["Overweight", "BackPain"]].copy()
overweight_map = {1 : "Overweight", 0: "NotOverweight"}
arthritis_map = {1 : "Arthritis", 0: "No Arthritis"}
temp_df["Overweight"] = temp_df["Overweight"].map(overweight_map)
temp_df["BackPain"] = temp_df["BackPain"].map(arthritis_map)
```

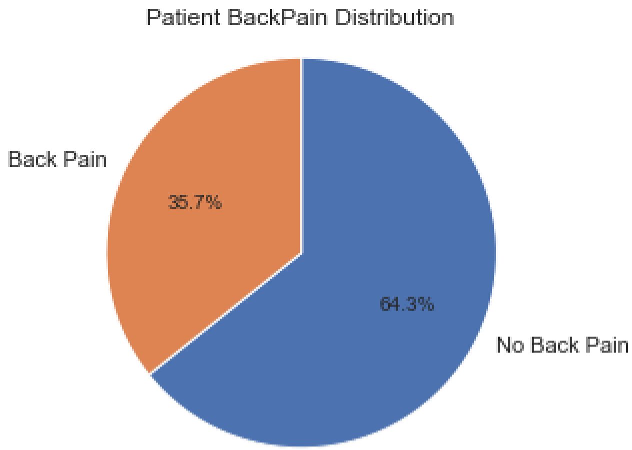
```
mosaic(temp_df, ["Overweight", "BackPain"])
plt.suptitle("Relationship of Overweight vs Arthritis");
```

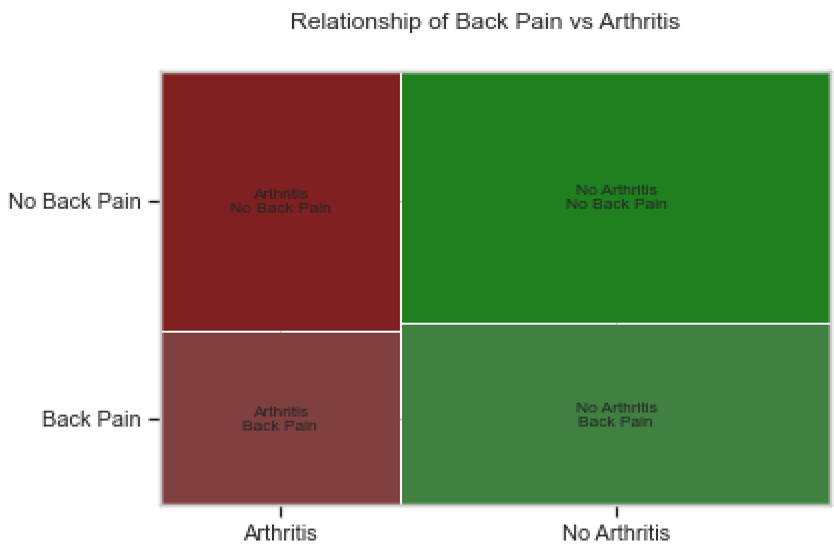




In [29]:
```
#Univariate and bivariate distribution of BackPain
# TOP plot: Univariate exploration of arthritis
plt.title("Patient BackPain Distribution")
back_pain_counts = df["BackPain"].value_counts().sort_index()
plt.pie(arthritis_counts, labels=["No Back Pain", "Back Pain"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# BOTTOM plot: Bivariate exploration of arthritis vs back_pain
temp_df = df[["Arthritis", "BackPain"]].copy()
```

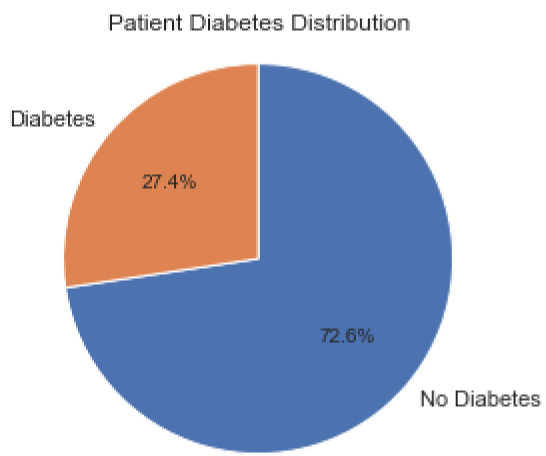
```
back_pain_map = {1 : "Back Pain", 0: "No Back Pain"}
arthritis_map = {1 : "Arthritis", 0: "No Arthritis"}
temp_df["BackPain"] = temp_df["BackPain"].map(back_pain_map)
temp_df["Arthritis"] = temp_df["Arthritis"].map(arthritis_map)
mosaic(temp_df, ["Arthritis", "BackPain"])
plt.suptitle("Relationship of Back Pain vs Arthritis");
```

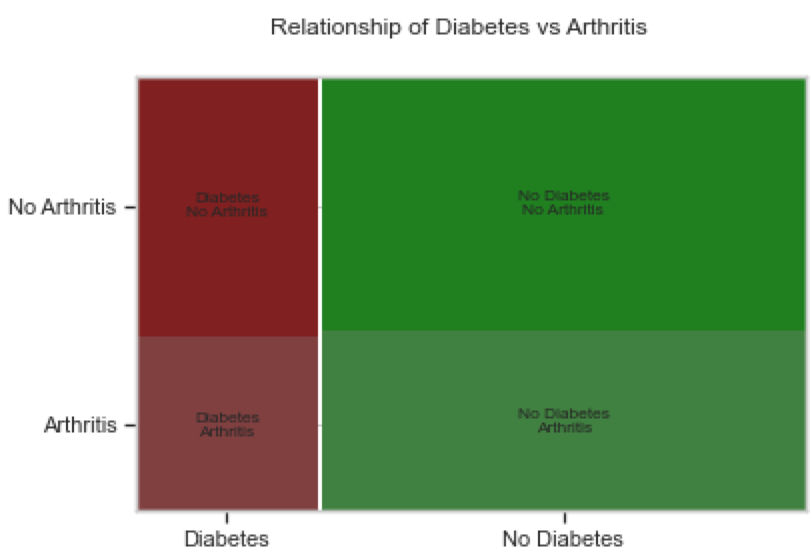




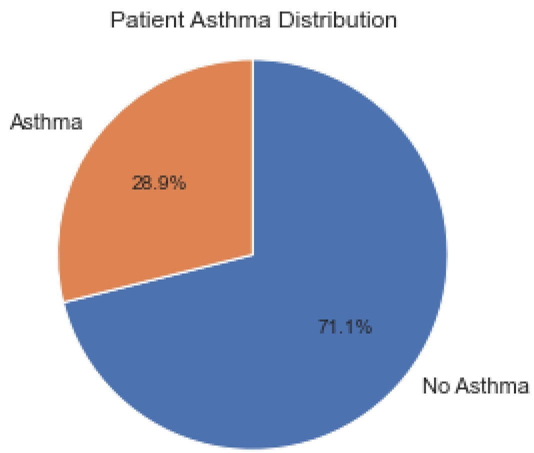
In [30]:
```
#Univariate and bivariate distribution of Diabetes
# TOP plot: Univariate exploration of diabetes
plt.title("Patient Diabetes Distribution")
diabetes_counts = df["Diabetes"].value_counts().sort_index()
plt.pie(diabetes_counts, labels=["No Diabetes", "Diabetes"], autopct='%1.1f%%', startangle=90, counterclock = False)
```

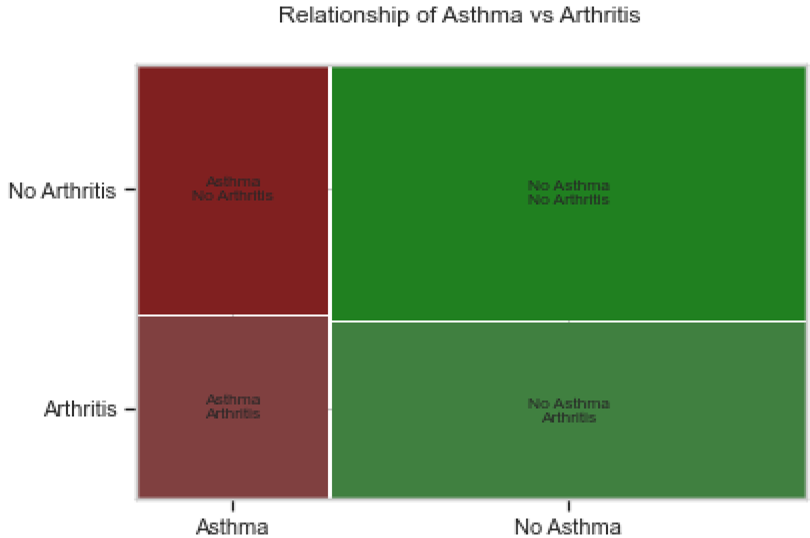
```
plt.axis('square');

# BOTTOM plot: Bivariate exploration of diabetes vs arthritis
temp_df = df[["Diabetes", "BackPain"]].copy()
diabetes_map = {1 : "Diabetes", 0: "No Diabetes"}
arthritis_map = {1 : "Arthritis", 0: "No Arthritis"}
temp_df["Diabetes"] = temp_df["Diabetes"].map(diabetes_map)
temp_df["BackPain"] = temp_df["BackPain"].map(arthritis_map)
mosaic(temp_df, ["Diabetes", "BackPain"])
plt.suptitle("Relationship of Diabetes vs Arthritis");
```



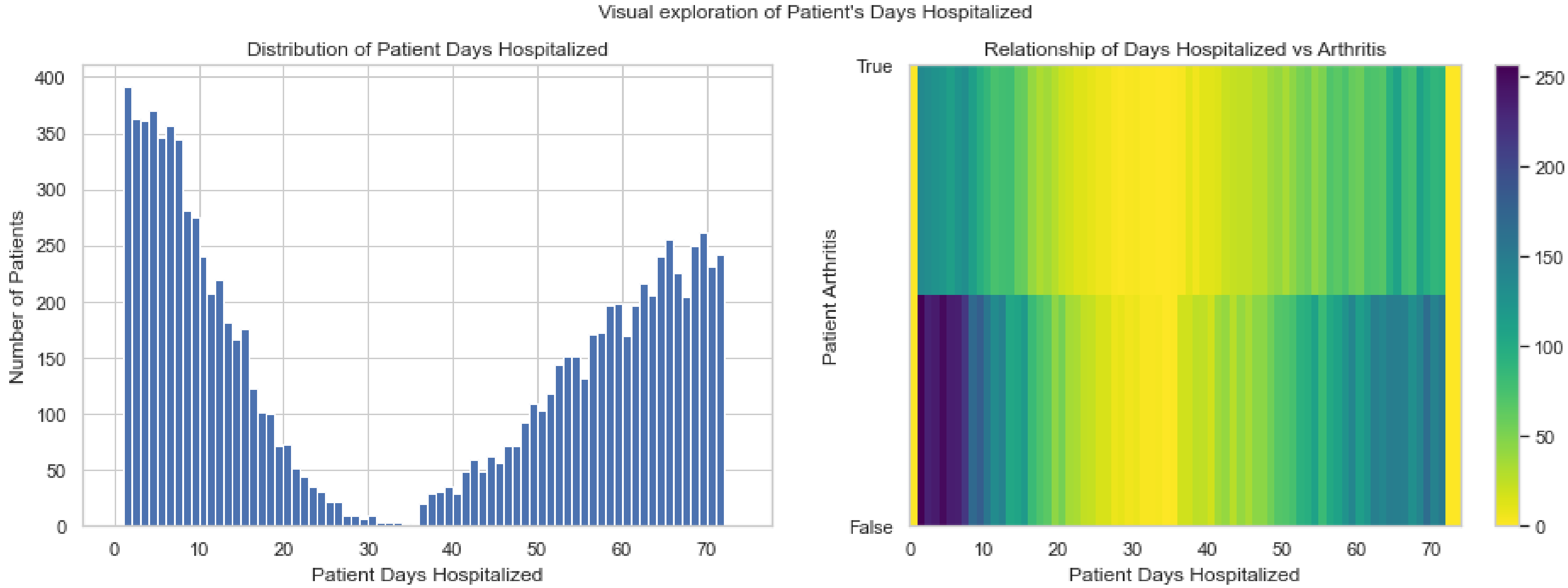


In [31]:
```
#Univariate and bivariate distribution of Asthma
# TOP plot: Univariate exploration of asthma
plt.title("Patient Asthma Distribution")
asthma_counts = df["Asthma"].value_counts()
plt.pie(asthma_counts, labels=["No Asthma", "Asthma"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# BOTTOM plot: Bivariate exploration of asthma vs arthritis
temp_df = df[["Asthma", "BackPain"]].copy()
asthma_map = {1 : "Asthma", 0: "No Asthma"}
arthritis_map = {1 : "Arthritis", 0: "No Arthritis"}
temp_df["Asthma"] = temp_df["Asthma"].map(asthma_map)
temp_df["BackPain"] = temp_df["BackPain"].map(arthritis_map)
mosaic(temp_df, ["Asthma", "BackPain"])
plt.suptitle("Relationship of Asthma vs Arthritis");
```





In [32]:
```
#Univariate and bivariate distribution of Days Hospitalized
plt.figure(figsize = [16,5])
plt.suptitle("Visual exploration of Patient's Days Hospitalized")

# LEFT plot: Univariate exploration of days_hospitalized
plt.subplot(1, 2, 1)
plt.title('Distribution of Patient Days Hospitalized')
bins = np.arange(0, 75, 1)
plt.hist(data=regress_df, x="days_hospitalized", bins=bins)
plt.xlabel('Patient Days Hospitalized')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of days_hospitalized vs arthritis
plt.subplot(1, 2, 2)
plt.title("Relationship of Days Hospitalized vs Arthritis")
bins_y = np.arange(0, 1.25, 0.5)
plt.hist2d(data= regress_df, x="days_hospitalized", y="arthritis", bins=[bins, bins_y], cmap= "viridis_r")
plt.colorbar()
plt.xlabel("Patient Days Hospitalized")
plt.ylabel("Patient Arthritis")
plt.yticks([0,1], ["False", "True"]);
```



In [33]:
```
# Save dataframe to CSV
df.to_csv('d208task2_full_clean.csv', index=False)

# Save dataframe to CSV
regress_df.to_csv('d208task2_red_clean.csv', index=False)
```

In [34]:
```
# Check for VIF to determine if variables should be eliminated due to high multicolinearity
# Selecting the features for VIF calculation
X = regress_df[["age", "gender_male", "gender_nonbinary", "vit_d_level", "initial_admit_observ", "initial_admit_emerg",

# Calculating VIF for each feature
vif_df = pd.DataFrame()
vif_df["feature"] = X.columns
vif_df["vif"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

print(vif_df)
```

```
                 feature        vif
0                    age   7.241869
1            gender_male   1.925995
2       gender_nonbinary   1.043023
3            vit_d_level  16.271274
4   initial_admit_observ   1.937244
5    initial_admit_emerg   2.958304
6                high_bp   1.686147
7          comp_risk_low   1.616292
8       comp_risk_medium   2.314051
9             overweight   3.371494
10             back_pain   1.691642
11              diabetes   1.368503
12                asthma   1.404802
13     days_hospitalized   2.671751
```

In [35]:

```
#Create the Initial Logistic Regression model
y = regress_df.arthritis
X = regress_df[["age", "gender_male", "gender_nonbinary", "vit_d_level", "initial_admit_observ", "initial_admit_emerg",
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651068
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9985
Method:                           MLE   Df Model:                           14
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001288
Time:                        19:14:59   Log-Likelihood:                -6510.7
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                    0.2675
========================================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------
age                      0.0008      0.001      0.747      0.455      -0.001       0.003
gender_male              0.0396      0.042      0.936      0.349      -0.043       0.122
gender_nonbinary         0.1982      0.143      1.386      0.166      -0.082       0.478
vit_d_level          -2.453e-05      0.010     -0.002      0.998      -0.020       0.020
initial_admit_observ    -0.0007      0.059     -0.012      0.991      -0.117       0.116
initial_admit_emerg     -0.0005      0.051     -0.010      0.992      -0.101       0.100
high_bp                  0.0307      0.042      0.723      0.470      -0.053       0.114
comp_risk_low            0.0752      0.058      1.295      0.195      -0.039       0.189
comp_risk_medium         0.1020      0.048      2.137      0.033       0.008       0.196
overweight               0.0186      0.046      0.404      0.686      -0.072       0.109
back_pain               -0.0822      0.043     -1.931      0.053      -0.166       0.001
diabetes                 0.0422      0.047      0.902      0.367      -0.049       0.134
asthma                  -0.0301      0.046     -0.652      0.515      -0.121       0.060
days_hospitalized        0.0015      0.001      1.899      0.058   -4.82e-05       0.003
const                   -0.7592      0.208     -3.659      0.000      -1.166      -0.353
========================================================================================
```

In [36]:

```
# Check for VIF to see if variables should be eliminated due to high multicolinearity
X = regress_df[["age", "gender_male", "gender_nonbinary", "vit_d_level", "initial_admit_observ", "initial_admit_emerg",

vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                  feature        VIF
0                     age   7.241869
1             gender_male   1.925995
2        gender_nonbinary   1.043023
3             vit_d_level  16.271274
4     initial_admit_observ   1.937244
5      initial_admit_emerg   2.958304
6                 high_bp   1.686147
7            comp_risk_low   1.616292
8         comp_risk_medium   2.314051
9               overweight   3.371494
10               back_pain   1.691642
11                diabetes   1.368503
12                  asthma   1.404802
13       days_hospitalized   2.671751
```

In [37]:

```
# Eliminated vit_d_level (VIF = 16.271274), rerunning analysis to see if any VIF still above 10
X = regress_df[["age", "gender_male", "gender_nonbinary", "initial_admit_observ", "initial_admit_emerg", "high_bp", "co

vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                  feature       VIF
0                     age  5.106574
1             gender_male  1.841550
2        gender_nonbinary  1.039726
3     initial_admit_observ  1.777416
4      initial_admit_emerg  2.587708
5                 high_bp  1.644376
6            comp_risk_low  1.548105
7         comp_risk_medium  2.162291
8               overweight  3.028072
9                back_pain  1.658025
10                diabetes  1.350051
11                  asthma  1.385518
12       days_hospitalized  2.493187
```

In [38]:

```
# BACKWARD ELIMINATION # 1: Seek highest p-value above 0.10
y = regress_df.arthritis
X = regress_df[["age", "gender_male", "gender_nonbinary", "initial_admit_observ", "initial_admit_emerg", "high_bp", "co
```

```
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651068
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9986
Method:                           MLE   Df Model:                           13
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001288
Time:                        19:14:59   Log-Likelihood:                -6510.7
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                    0.2091
========================================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------
age                      0.0008      0.001      0.747      0.455      -0.001       0.003
gender_male              0.0396      0.042      0.936      0.349      -0.043       0.122
gender_nonbinary         0.1982      0.143      1.386      0.166      -0.082       0.478
initial_admit_observ    -0.0007      0.059     -0.012      0.991      -0.117       0.116
initial_admit_emerg     -0.0005      0.051     -0.010      0.992      -0.101       0.100
high_bp                  0.0307      0.042      0.723      0.470      -0.053       0.114
comp_risk_low            0.0752      0.058      1.295      0.195      -0.039       0.189
comp_risk_medium         0.1020      0.048      2.137      0.033       0.008       0.196
overweight               0.0186      0.046      0.404      0.686      -0.072       0.109
back_pain               -0.0822      0.043     -1.931      0.053      -0.166       0.001
diabetes                 0.0422      0.047      0.903      0.367      -0.049       0.134
asthma                  -0.0301      0.046     -0.652      0.515      -0.121       0.060
days_hospitalized        0.0015      0.001      1.899      0.058   -4.82e-05       0.003
const                   -0.7597      0.094     -8.119      0.000      -0.943      -0.576
========================================================================================
```

In [39]:

```
# BACKWARD ELIMINATION # 2: Seek highest p-value above 0.10 (eliminated initial_admit_emerg, p-value of 0.992)
y = regress_df.arthritis
X = regress_df[["age", "gender_male", "gender_nonbinary", "initial_admit_observ", "high_bp", "comp_risk_low", "comp_ris
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651068
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9987
Method:                           MLE   Df Model:                           12
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001288
Time:                        19:14:59   Log-Likelihood:                -6510.7
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                    0.1577
========================================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------
age                      0.0008      0.001      0.747      0.455      -0.001       0.003
gender_male              0.0396      0.042      0.936      0.349      -0.043       0.122
gender_nonbinary         0.1982      0.143      1.386      0.166      -0.082       0.478
initial_admit_observ    -0.0004      0.049     -0.008      0.994      -0.096       0.095
high_bp                  0.0307      0.042      0.723      0.470      -0.053       0.114
comp_risk_low            0.0752      0.058      1.295      0.195      -0.039       0.189
comp_risk_medium         0.1020      0.048      2.137      0.033       0.008       0.196
overweight               0.0186      0.046      0.404      0.686      -0.072       0.109
back_pain               -0.0822      0.043     -1.931      0.053      -0.166       0.001
diabetes                 0.0422      0.047      0.903      0.367      -0.049       0.134
asthma                  -0.0301      0.046     -0.652      0.515      -0.121       0.060
days_hospitalized        0.0015      0.001      1.900      0.057    -4.8e-05       0.003
const                   -0.7600      0.087     -8.776      0.000      -0.930      -0.590
========================================================================================
```

In [40]:

```
# BACKWARD ELIMINATION # 3: Seek highest p-value above 0.10 (eliminated initial_admit_observ, p-value of 0.994)
y = regress_df.arthritis
X = regress_df[["age", "gender_male", "gender_nonbinary", "high_bp", "comp_risk_low", "comp_risk_medium", "overweight",
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651068
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9988
Method:                           MLE   Df Model:                           11
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001288
Time:                        19:14:59   Log-Likelihood:                -6510.7
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                    0.1142
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
age                   0.0008      0.001      0.747      0.455      -0.001       0.003
gender_male           0.0396      0.042      0.937      0.349      -0.043       0.122
gender_nonbinary      0.1982      0.143      1.386      0.166      -0.082       0.478
high_bp               0.0307      0.042      0.723      0.470      -0.053       0.114
comp_risk_low         0.0752      0.058      1.295      0.195      -0.039       0.189
comp_risk_medium      0.1020      0.048      2.138      0.033       0.008       0.196
overweight            0.0186      0.046      0.404      0.686      -0.072       0.109
back_pain            -0.0822      0.043     -1.931      0.053      -0.166       0.001
diabetes              0.0422      0.047      0.903      0.367      -0.049       0.134
asthma               -0.0301      0.046     -0.652      0.515      -0.121       0.060
days_hospitalized     0.0015      0.001      1.900      0.057   -4.8e-05       0.003
const                -0.7601      0.086     -8.853      0.000      -0.928      -0.592
=====================================================================================
```

In [41]:
```python
# BACKWARD ELIMINATION # 4: Seek highest p-value above 0.10 (eliminated overweight, p-value of 0.686)
y = regress_df.arthritis
X = regress_df[["age", "gender_male", "gender_nonbinary", "high_bp", "comp_risk_low", "comp_risk_medium", "back_pain",
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651076
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9989
Method:                           MLE   Df Model:                           10
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001275
Time:                        19:15:00   Log-Likelihood:                -6510.8
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                   0.08303
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
age                   0.0008      0.001      0.743      0.457      -0.001       0.003
gender_male           0.0395      0.042      0.936      0.349      -0.043       0.122
gender_nonbinary      0.1984      0.143      1.388      0.165      -0.082       0.479
high_bp               0.0312      0.042      0.734      0.463      -0.052       0.114
comp_risk_low         0.0752      0.058      1.294      0.196      -0.039       0.189
comp_risk_medium      0.1023      0.048      2.144      0.032       0.009       0.196
back_pain            -0.0821      0.043     -1.927      0.054      -0.165       0.001
diabetes              0.0420      0.047      0.900      0.368      -0.050       0.134
asthma               -0.0298      0.046     -0.647      0.518      -0.120       0.061
days_hospitalized     0.0015      0.001      1.895      0.058   -5.13e-05       0.003
const                -0.7470      0.079     -9.401      0.000      -0.903      -0.591
=====================================================================================
```

In [42]:

```
# BACKWARD ELIMINATION # 5: Seek highest p-value above 0.10 (eliminated asthma, p-value of 0.518)
y = regress_df.arthritis
X = regress_df[["age", "gender_male", "gender_nonbinary", "high_bp", "comp_risk_low", "comp_risk_medium", "back_pain",
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651097
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9990
Method:                           MLE   Df Model:                            9
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001243
Time:                        19:15:00   Log-Likelihood:                -6511.0
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                   0.06265
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
age                   0.0007      0.001      0.737      0.461      -0.001       0.003
gender_male           0.0394      0.042      0.932      0.351      -0.043       0.122
gender_nonbinary      0.1980      0.143      1.385      0.166      -0.082       0.478
high_bp               0.0310      0.042      0.729      0.466      -0.052       0.114
comp_risk_low         0.0749      0.058      1.289      0.197      -0.039       0.189
comp_risk_medium      0.1020      0.048      2.138      0.033       0.008       0.196
back_pain            -0.0824      0.043     -1.937      0.053      -0.166       0.001
diabetes              0.0415      0.047      0.889      0.374      -0.050       0.133
days_hospitalized     0.0015      0.001      1.905      0.057   -4.39e-05       0.003
const                -0.7549      0.079     -9.613      0.000      -0.909      -0.601
=====================================================================================
```

In [43]:

```python
# BACKWARD ELIMINATION # 6: Seek highest p-value above 0.10 (eliminated age, p-value of 0.461)
y = regress_df.arthritis
X = regress_df[["gender_male", "gender_nonbinary", "high_bp", "comp_risk_low", "comp_risk_medium", "back_pain", "diabet
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651124
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9991
Method:                           MLE   Df Model:                            8
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001201
Time:                        19:15:00   Log-Likelihood:                -6511.2
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                   0.04744
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
gender_male           0.0389      0.042      0.921      0.357      -0.044       0.122
gender_nonbinary      0.1980      0.143      1.386      0.166      -0.082       0.478
high_bp               0.0312      0.042      0.735      0.462      -0.052       0.114
comp_risk_low         0.0747      0.058      1.287      0.198      -0.039       0.189
comp_risk_medium      0.1018      0.048      2.133      0.033       0.008       0.195
back_pain            -0.0818      0.043     -1.922      0.055      -0.165       0.002
diabetes              0.0416      0.047      0.892      0.373      -0.050       0.133
days_hospitalized     0.0015      0.001      1.917      0.055   -3.45e-05       0.003
const                -0.7152      0.057    -12.509      0.000      -0.827      -0.603
=====================================================================================
```

In [44]:

```
# BACKWARD ELIMINATION # 7: Seek highest p-value above 0.10 (eliminated gender_male, p-value of 0.357)
y = regress_df.arthritis
X = regress_df[["gender_nonbinary", "high_bp", "comp_risk_low", "comp_risk_medium", "back_pain", "diabetes", "days_hosp
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651167
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9992
Method:                           MLE   Df Model:                            7
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001136
Time:                        19:15:00   Log-Likelihood:                -6511.7
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                   0.03842
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
gender_nonbinary      0.1790      0.141      1.266      0.206      -0.098       0.456
high_bp               0.0315      0.042      0.741      0.459      -0.052       0.115
comp_risk_low         0.0746      0.058      1.285      0.199      -0.039       0.188
comp_risk_medium      0.1021      0.048      2.139      0.032       0.009       0.196
back_pain            -0.0823      0.043     -1.934      0.053      -0.166       0.001
diabetes              0.0416      0.047      0.890      0.374      -0.050       0.133
days_hospitalized     0.0015      0.001      1.924      0.054   -2.86e-05       0.003
const                -0.6964      0.053    -13.050      0.000      -0.801      -0.592
=====================================================================================
```

In [45]:

```python
# BACKWARD ELIMINATION # 8: Seek highest p-value above 0.10 (eliminated high_bp, p-value of 0.459)
y = regress_df.arthritis
X = regress_df[["gender_nonbinary", "comp_risk_low", "comp_risk_medium", "back_pain", "diabetes", "days_hospitalized"]]
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651194
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9993
Method:                           MLE   Df Model:                            6
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001094
Time:                        19:15:00   Log-Likelihood:                -6511.9
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                   0.02678
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
gender_nonbinary      0.1806      0.141      1.277      0.202      -0.097       0.458
comp_risk_low         0.0736      0.058      1.268      0.205      -0.040       0.187
comp_risk_medium      0.1021      0.048      2.140      0.032       0.009       0.196
back_pain            -0.0822      0.043     -1.931      0.053      -0.166       0.001
diabetes              0.0414      0.047      0.886      0.376      -0.050       0.133
days_hospitalized     0.0015      0.001      1.919      0.055   -3.21e-05       0.003
const                -0.6832      0.050    -13.585      0.000      -0.782      -0.585
=====================================================================================
```

In [46]:

```
# BACKWARD ELIMINATION # 9: Seek highest p-value above 0.10 (eliminated diabetes, p-value of 0.376)
y = regress_df.arthritis
X = regress_df[["gender_nonbinary", "comp_risk_low", "comp_risk_medium", "back_pain", "days_hospitalized"]].assign(cons
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651233
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9994
Method:                           MLE   Df Model:                            5
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:                0.001034
Time:                        19:15:01   Log-Likelihood:                -6512.3
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                   0.01923
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
gender_nonbinary      0.1803      0.141      1.275      0.202      -0.097       0.457
comp_risk_low         0.0739      0.058      1.274      0.203      -0.040       0.188
comp_risk_medium      0.1022      0.048      2.142      0.032       0.009       0.196
back_pain            -0.0827      0.043     -1.943      0.052      -0.166       0.001
days_hospitalized     0.0015      0.001      1.917      0.055   -3.39e-05       0.003
const                -0.6716      0.049    -13.833      0.000      -0.767      -0.576
=====================================================================================
```

In [47]:
```
# BACKWARD ELIMINATION # 10: Seek highest p-value above 0.10 (eliminated comp_risk_low, p-value of 0.203)
y = regress_df.arthritis
X = regress_df[["gender_nonbinary", "comp_risk_medium", "back_pain", "days_hospitalized"]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651314
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9995
Method:                           MLE   Df Model:                            4
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:               0.0009100
Time:                        19:15:01   Log-Likelihood:                -6513.1
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                   0.01839
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
gender_nonbinary      0.1784      0.141      1.262      0.207      -0.099       0.456
comp_risk_medium      0.0734      0.042      1.750      0.080      -0.009       0.156
back_pain            -0.0817      0.043     -1.922      0.055      -0.165       0.002
days_hospitalized     0.0015      0.001      1.937      0.053    -1.8e-05       0.003
const                -0.6437      0.043    -14.890      0.000      -0.728      -0.559
=====================================================================================
```

In [48]:

```
# BACKWARD ELIMINATION # 11: Seek highest p-value above 0.10 (eliminated gender_nonbinary, p-value of 0.207)
y = regress_df.arthritis
X = regress_df[["comp_risk_medium", "back_pain", "days_hospitalized"]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651393
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9996
Method:                           MLE   Df Model:                            3
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:               0.0007894
Time:                        19:15:01   Log-Likelihood:                -6513.9
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                   0.01624
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
comp_risk_medium      0.0732      0.042      1.747      0.081      -0.009       0.155
back_pain            -0.0807      0.043     -1.897      0.058      -0.164       0.003
days_hospitalized     0.0015      0.001      1.942      0.052   -1.44e-05       0.003
const                -0.6403      0.043    -14.843      0.000      -0.725      -0.556
=====================================================================================
```

In [49]:

```
# All p-values for independent variables are < 0.10, this is the final reduced model
y = regress_df.arthritis
X = regress_df[["comp_risk_medium", "back_pain", "days_hospitalized"]].assign(const=1)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.651393
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:              arthritis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9996
Method:                           MLE   Df Model:                            3
Date:                Sun, 22 Oct 2023   Pseudo R-squ.:               0.0007894
Time:                        19:15:01   Log-Likelihood:                -6513.9
converged:                       True   LL-Null:                       -6519.1
Covariance Type:            nonrobust   LLR p-value:                   0.01624
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
comp_risk_medium      0.0732      0.042      1.747      0.081      -0.009       0.155
back_pain            -0.0807      0.043     -1.897      0.058      -0.164       0.003
days_hospitalized     0.0015      0.001      1.942      0.052   -1.44e-05       0.003
const                -0.6403      0.043    -14.843      0.000      -0.725      -0.556
=====================================================================================
```

In [55]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
final_matrix = confusion_matrix(y_test, y_pred)
print(final_matrix)
```

```
Accuracy of logistic regression classifier on test set: 0.66
[[1319    0]
 [ 681    0]]
```

In [51]:

```
result.params
```

Out[51]:

```
comp_risk_medium     0.073224
back_pain           -0.080667
days_hospitalized    0.001541
const               -0.640284
dtype: float64
```

In [52]:

```
# Calculate odds ratios for each coefficient
print(f"The odds ratio for comp_risk_medium is {round(np.exp(0.073224), 3)}. Given this, the change in odds for arthrit
print(f"The odds ratio for back_pain is {round(np.exp(0.080667), 3)}. Given this, the change in odds for arthritis is {
print(f"The odds ratio for days_hospitalized is {round(np.exp (0.001541), 3)}. Given this, the change in odds for arthr
```

```
The odds ratio for comp_risk_medium is 1.076. Given this, the change in odds for arthritis is 7.597
The odds ratio for back_pain is 1.084. Given this, the change in odds for arthritis is 8.401
The odds ratio for days_hospitalized is 1.002. Given this, the change in odds for arthritis is 0.154
```

In [ ]:

```

```