Case Study On Ashthma Drug Efficiency

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

df=pd.read_csv("UNMATCHED_PATIENTS.csv")
```

The info() method prints information about the DataFrame. The information contains the number of columns=21, column labels, column data types=int,float, memory usage=2.9mb, range index=18215 entries, and the number of cells in each column.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18215 entries, 0 to 18214
Data columns (total 21 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   patid                           18215 non-null  int64  
 1   index_age                       18215 non-null  int64  
 2   previous_asthma_drugs           18215 non-null  int64  
 3   total_pre_index_cannisters_365  18215 non-null  int64  
 4   post_index_exacerbations365     18215 non-null  int64  
 5   pneumonia                       18215 non-null  int64  
 6   sinusitis                       18215 non-null  int64  
 7   acute_bronchitis                18215 non-null  int64  
 8   acute_laryngitis                18215 non-null  int64  
 9   upper_respiratory_infection     18215 non-null  int64  
 10  gerd                            18215 non-null  int64  
 11  rhinitis                        18215 non-null  int64  
 12  adherence                       18215 non-null  float64
 13  total_pre_index_charge          18215 non-null  float64
 14  pre_asthma_days                 18215 non-null  int64  
 15  pre_asthma_charge               18215 non-null  float64
 16  pre_asthma_pharma_charge        18215 non-null  float64
 17  drug_s                          18215 non-null  int64  
 18  female                          18215 non-null  int64  
 19  log_charges                     18215 non-null  float64
 20  log_asthma_charge               18215 non-null  float64
dtypes: float64(6), int64(15)
memory usage: 2.9 MB
```

isnull().sum() returns the number of missing values in the data set. here, we have zero null values in the dataset

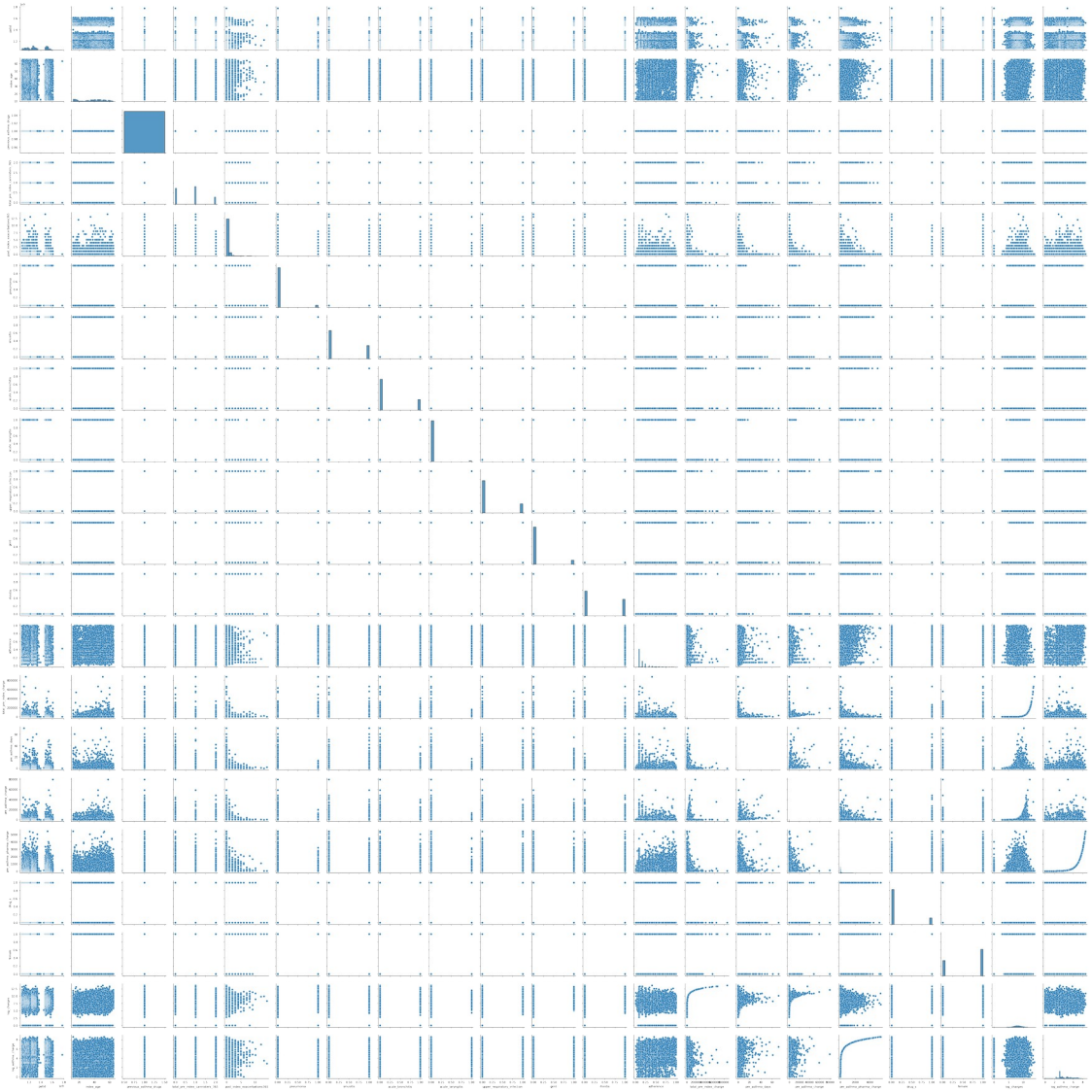
```
df.isnull().sum()
```

```
patid                              0
index_age                          0
previous_asthma_drugs              0
total_pre_index_cannisters_365     0
post_index_exacerbations365        0
pneumonia                          0
sinusitis                          0
acute_bronchitis                   0
acute_laryngitis                   0
upper_respiratory_infection        0
gerd                               0
rhinitis                           0
adherence                          0
total_pre_index_charge             0
pre_asthma_days                    0
pre_asthma_charge                  0
pre_asthma_pharma_charge           0
drug_s                             0
female                             0
log_charges                        0
log_asthma_charge                  0
dtype: int64
```

Pair plot is used to understand the best set of features to explain a relationship between two variables. It also helps to form some simple classification models by drawing some simple lines or make linear separation in our data-set. Here in this dataset we have no linear relationship with any of the feature and target column.

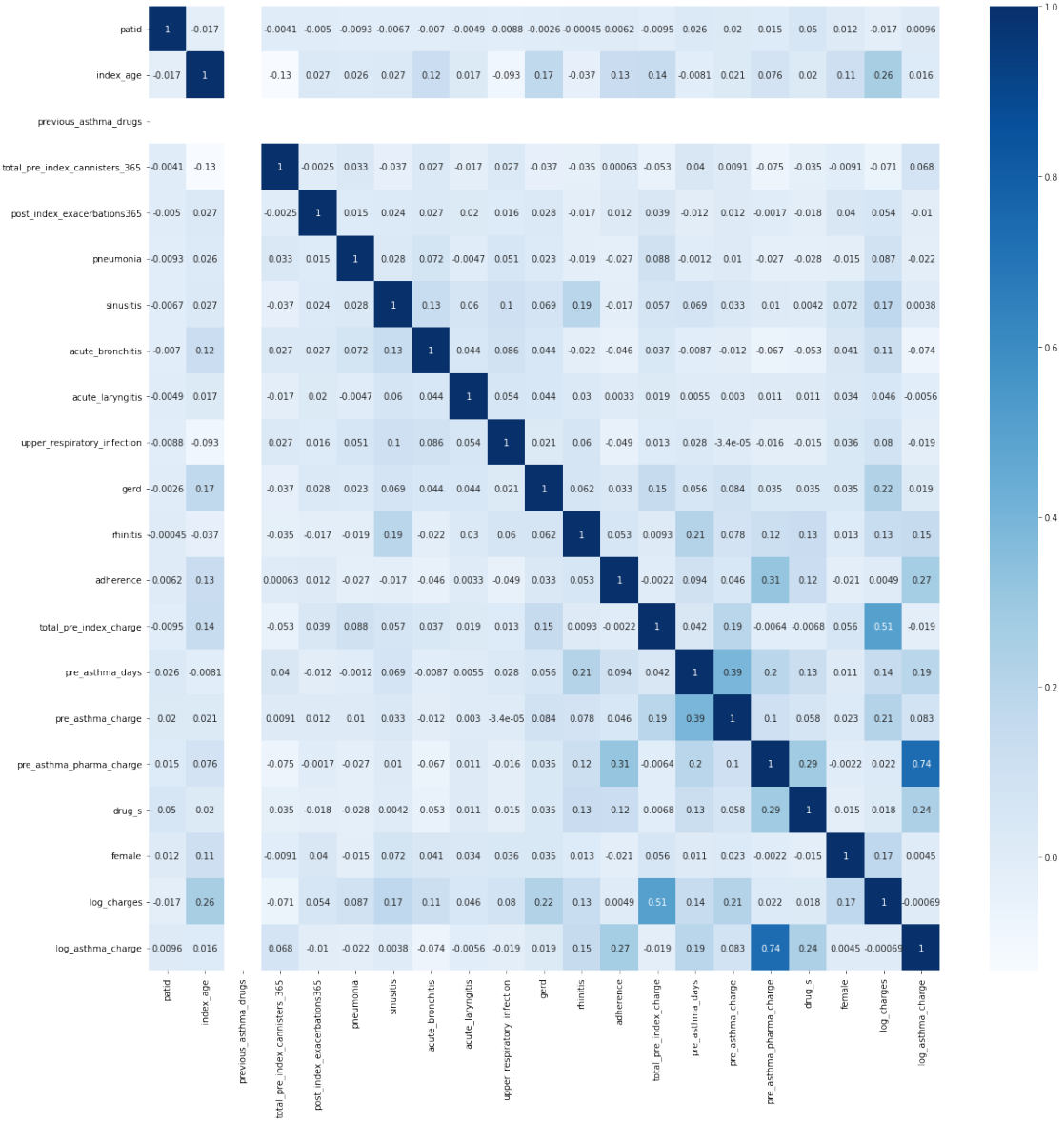
```
sns.pairplot(data=df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f771fbfd590>
```



heat maps are best used when something has changed either in your feature column and you want to understand how that affects useability i.e to your target column.

```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(),annot=True,cmap="Blues")
plt.show()
```



The describe() method returns description of the data in the DataFrame.

If the DataFrame contains numerical data, the description contains these information for each column:

count - The number of not-empty values. mean - The average (mean) value. std - The standard deviation. min - the minimum value. 25% - The 25% percentile. *50% - The 50% percentile*. 75% - The 75% percentile*. max - the maximum value.

```
df.describe()
```

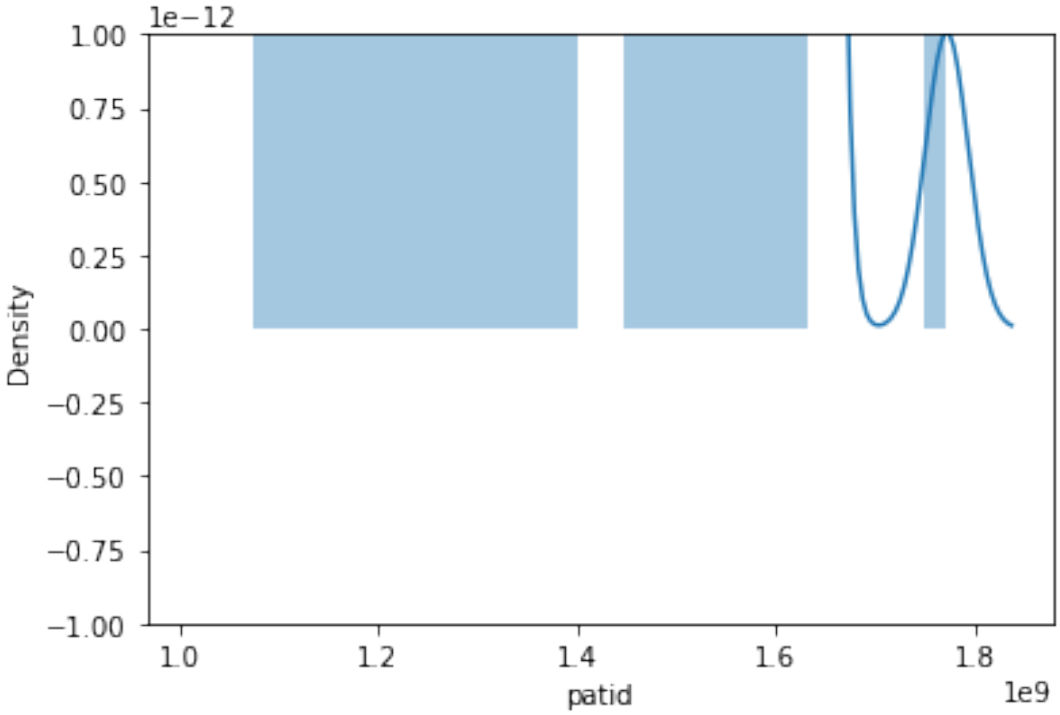
```
                patid     index_age  previous_asthma_drugs  \
count   1.821500e+04  18215.000000                18215.0
mean    1.317073e+09     38.304639                    1.0
std     1.545708e+08     15.218165                    0.0
```

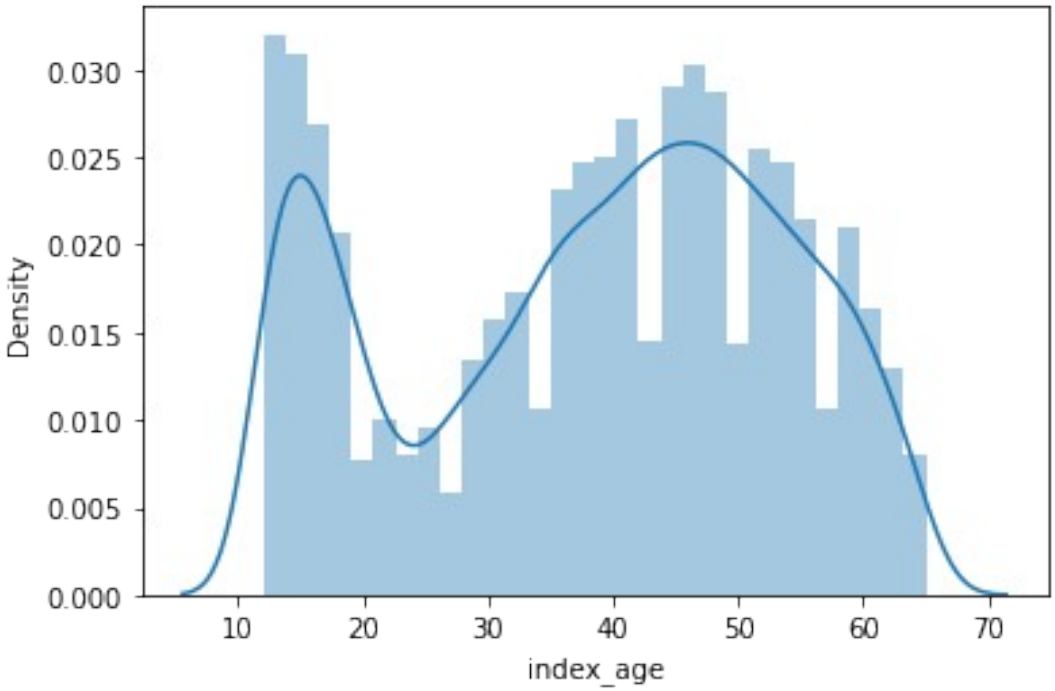
```
min    1.073754e+09     12.000000                   1.0
25%    1.183442e+09     26.000000                   1.0
50%    1.282389e+09     41.000000                   1.0
75%    1.493699e+09     50.000000                   1.0
max    1.771899e+09     65.000000                   1.0

       total_pre_index_cannisters_365  post_index_exacerbations365  \
count                    18215.000000                 18215.000000
mean                         0.793741                     0.174636
std                          0.723497                     0.633318
min                          0.000000                     0.000000
25%                          0.000000                     0.000000
50%                          1.000000                     0.000000
75%                          1.000000                     0.000000
max                          2.000000                    14.000000

          pneumonia     sinusitis  acute_bronchitis  acute_laryngitis
\
count  18215.000000  18215.000000      18215.000000      18215.000000

mean       0.050233      0.319682          0.260500          0.016854

std        0.218432      0.466366          0.438919          0.128729

min        0.000000      0.000000          0.000000          0.000000

25%        0.000000      0.000000          0.000000          0.000000

50%        0.000000      0.000000          0.000000          0.000000

75%        0.000000      1.000000          1.000000          0.000000

max        1.000000      1.000000          1.000000          1.000000


       upper_respiratory_infection  ...      rhinitis     adherence  \
count                 18215.000000  ...  18215.000000  18215.000000
mean                      0.221685  ...      0.400274      0.249604
std                       0.415392  ...      0.489967      0.227596
min                       0.000000  ...      0.000000      0.005450
25%                       0.000000  ...      0.000000      0.084469
50%                       0.000000  ...      0.000000      0.168937
75%                       0.000000  ...      1.000000      0.337875
max                       1.000000  ...      1.000000      1.000000

       total_pre_index_charge  pre_asthma_days  pre_asthma_charge  \
count            18215.000000     18215.000000       18215.000000
mean              8524.835020         1.414768         547.894775
std              21011.123711         2.457226        1940.953263
```

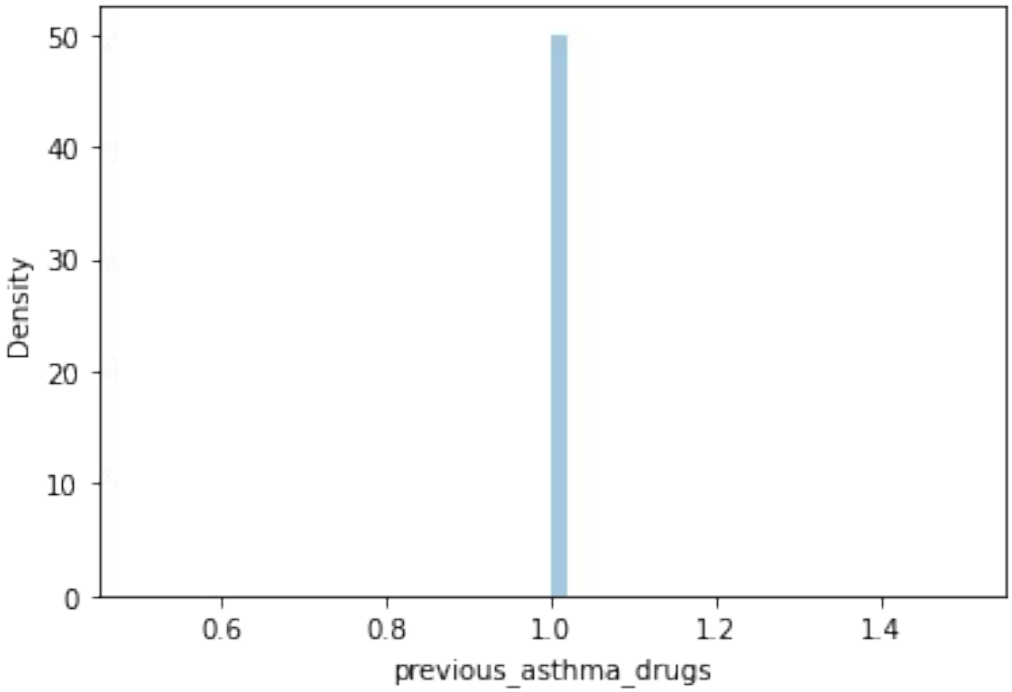
```
min                  1.000000         0.000000           0.000000
25%               1227.034917         0.000000           0.000000
50%               3154.941039         1.000000         130.000000
75%               8142.123760         2.000000         410.085000
max             875872.580590        71.000000       79280.910000

       pre_asthma_pharma_charge        drug_s        female
log_charges  \
count              18215.000000  18215.000000  18215.000000
18215.000000
mean                 244.820223      0.161405      0.632995
7.984189
std                  448.771943      0.367915      0.482001
1.649031
min                    1.000000      0.000000      0.000000
0.000000
25%                   21.410000      0.000000      0.000000
7.112356
50%                   46.300000      0.000000      1.000000
8.056725
75%                  229.560000      0.000000      1.000000
9.004806
max                 5463.140000      1.000000      1.000000
13.682976

       log_asthma_charge
count       18215.000000
mean            4.200143
std             1.710356
min             0.000000
25%             3.063858
50%             3.835142
75%             5.436164
max             8.605779

[8 rows x 21 columns]
```

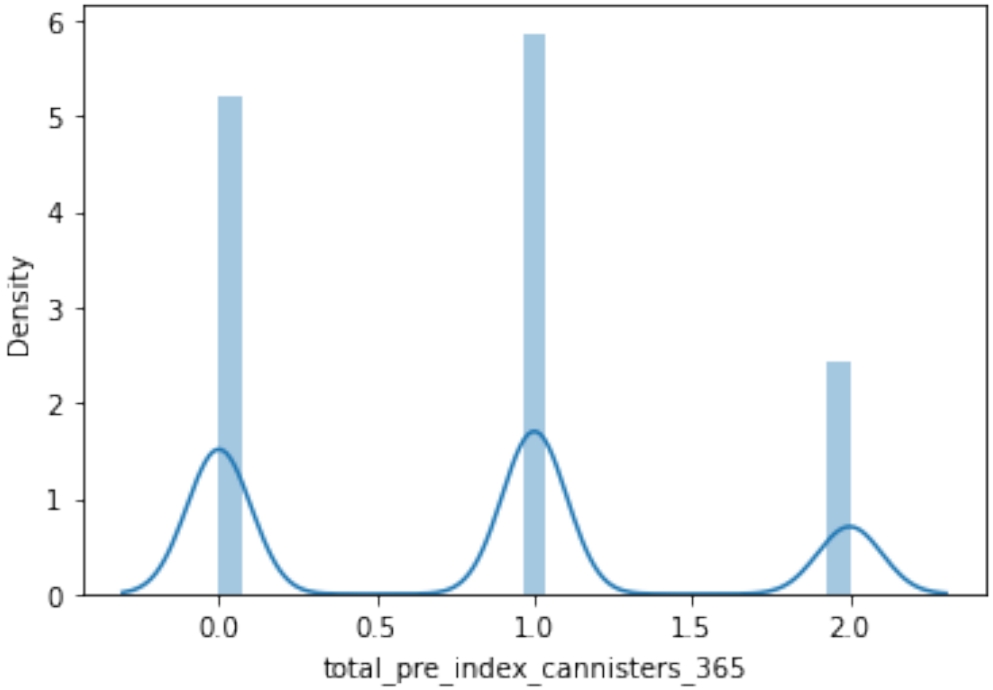
Distplot represents the overall distribution of continuous data variables you can see that there is a right skewness present in some columns and in some column the data is not normally distributed

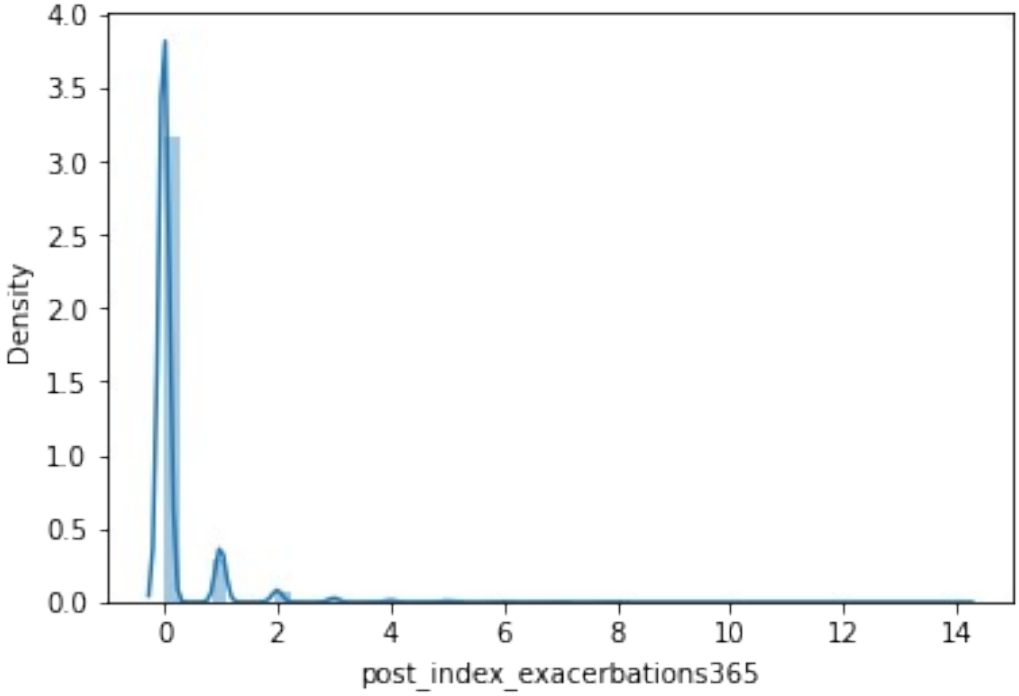
```
tuple=("patid","index_age","previous_asthma_drugs","total_pre_index_ca
nnisters_365","post_index_exacerbations365","pneumonia","sinusitis","a
cute_bronchitis","acute_laryngitis","upper_respiratory_infection","rhi
nitis","adherence","total_pre_index_charge","pre_asthma_days","pre_ast
hma_charge","pre_asthma_pharma_charge","drug_s","female","log_charges"
,"log_asthma_charge")
for col in tuple:
    sns.distplot(df[col], kde=True)
    plt.show()
```

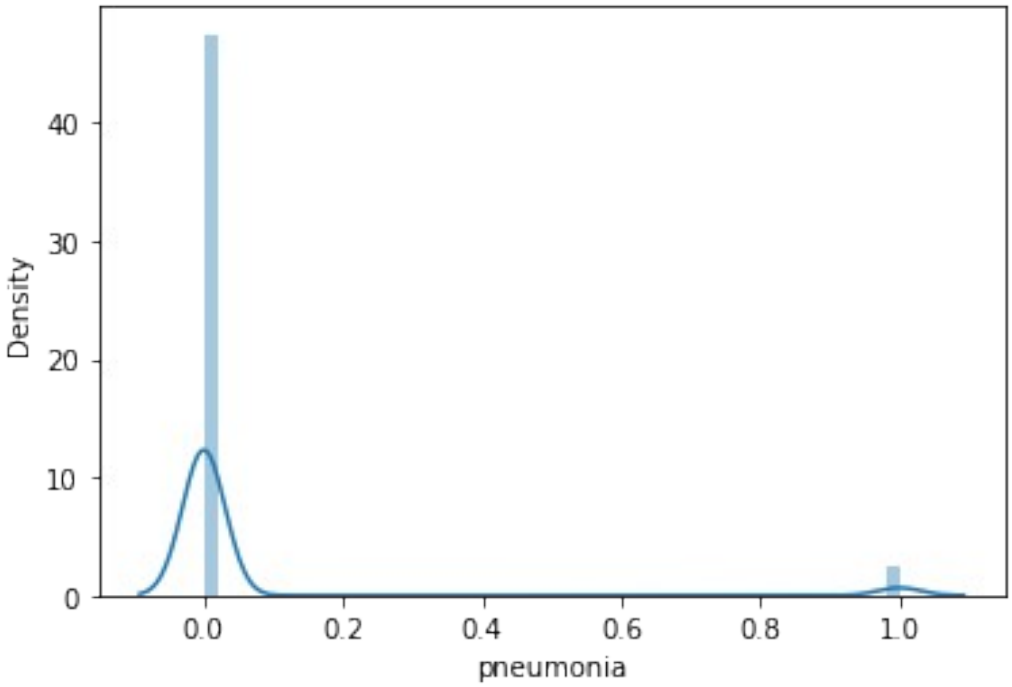

1e−12
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Density
1.0
1.2
1.4
1.6
1.8
1e9
patid

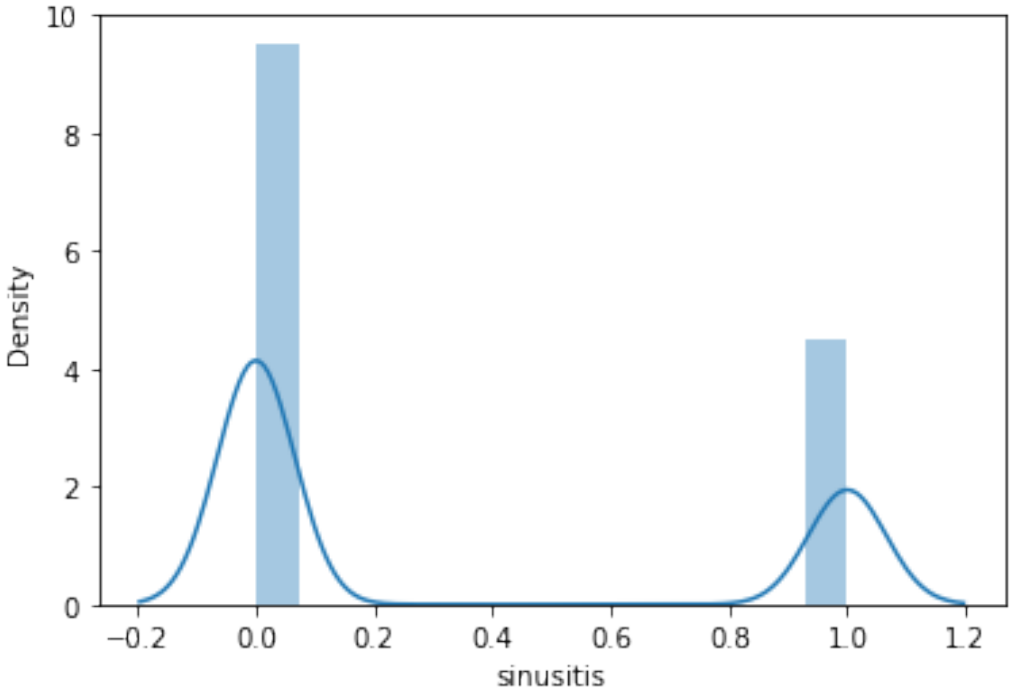

0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
10
20
30
40
50
60
70
index_age

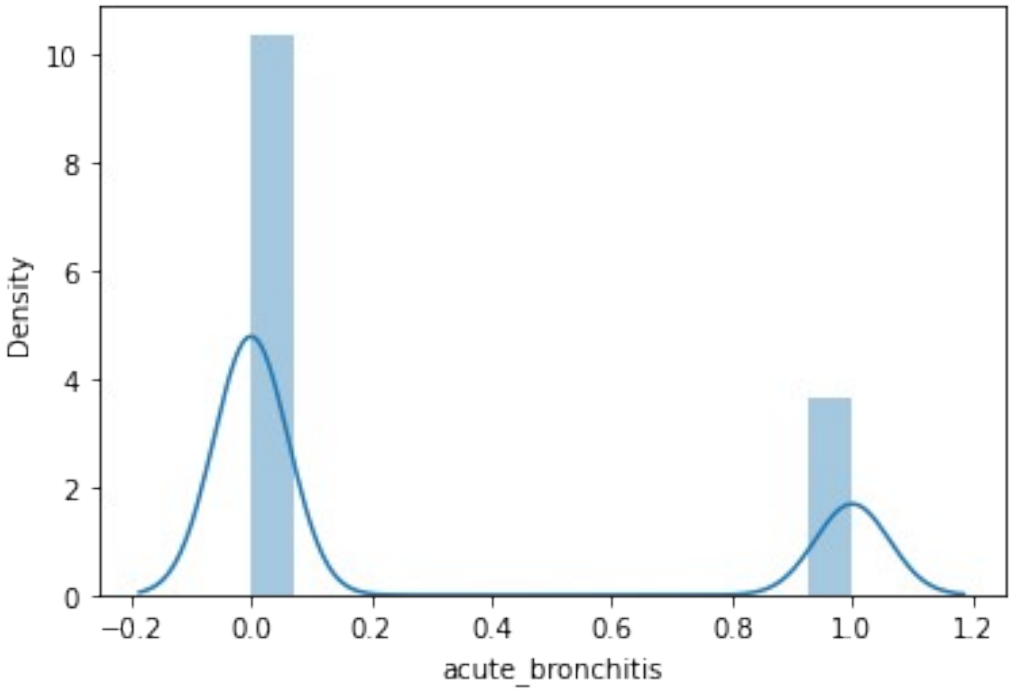

Density
50
40
30
20
10
0
0.6
0.8
1.0
1.2
1.4
previous_asthma_drugs

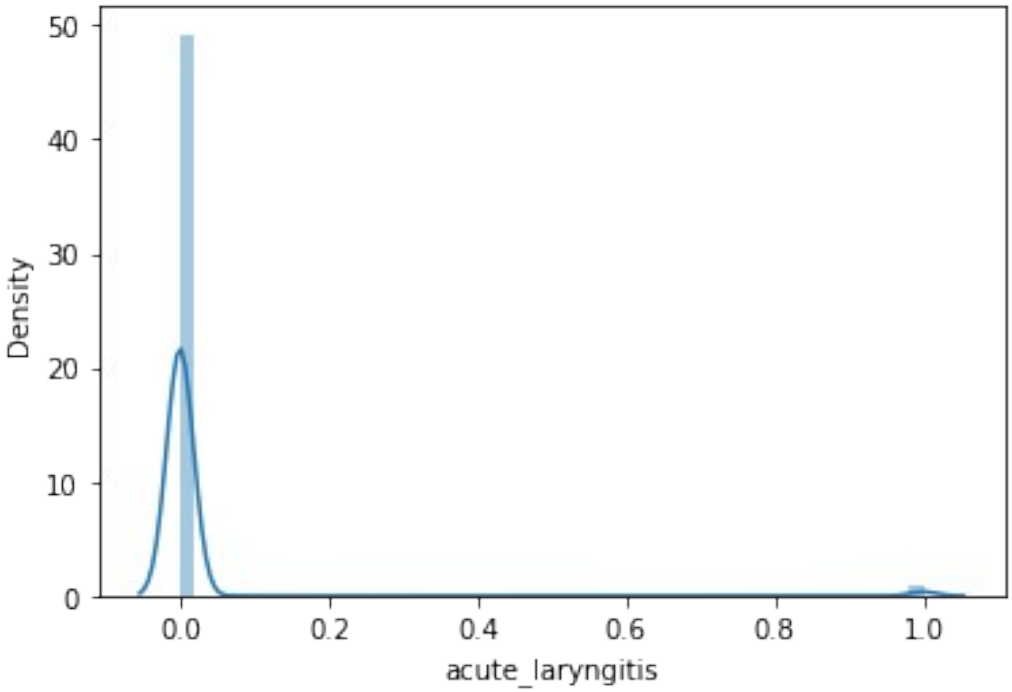

Density
6
5
4
3
2
1
0
0.0
0.5
1.0
1.5
2.0
total_pre_index_cannisters_365

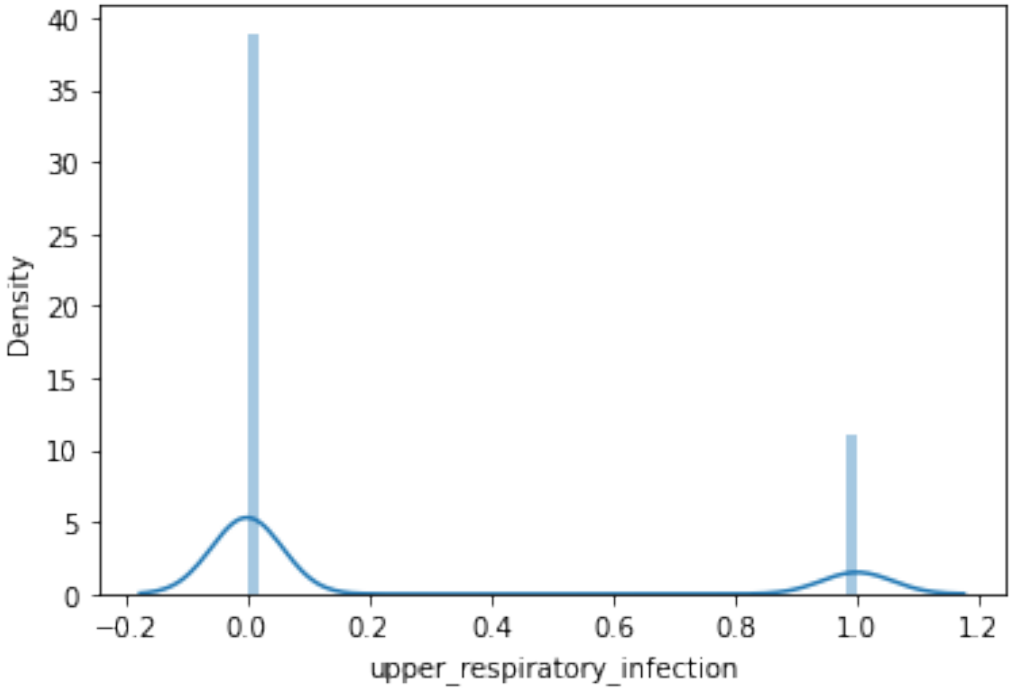

Density
post_index_exacerbations365

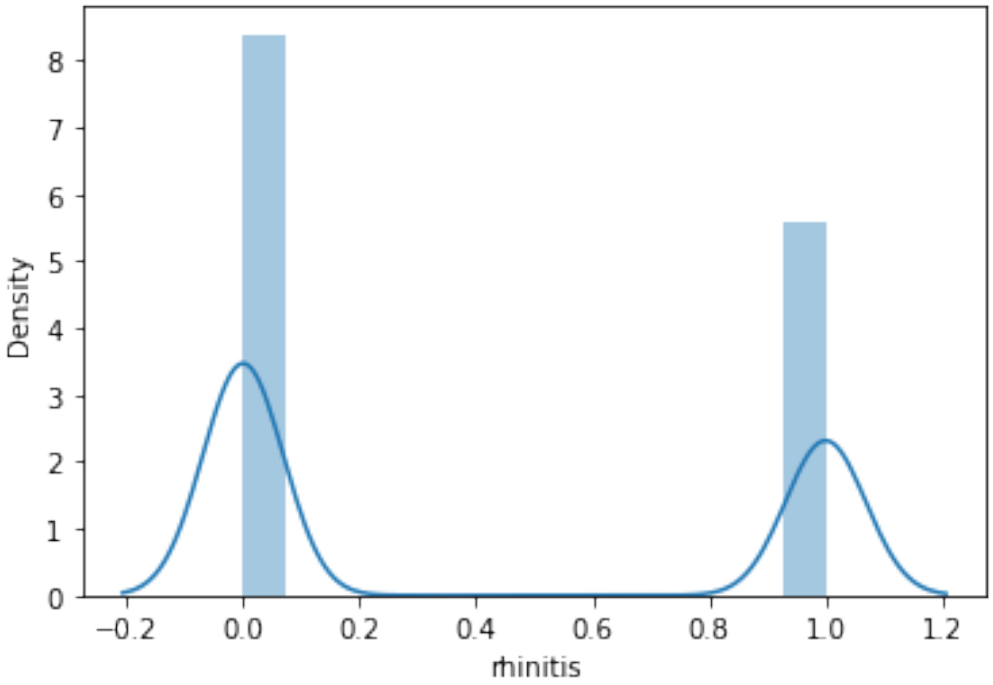

Density
pneumonia

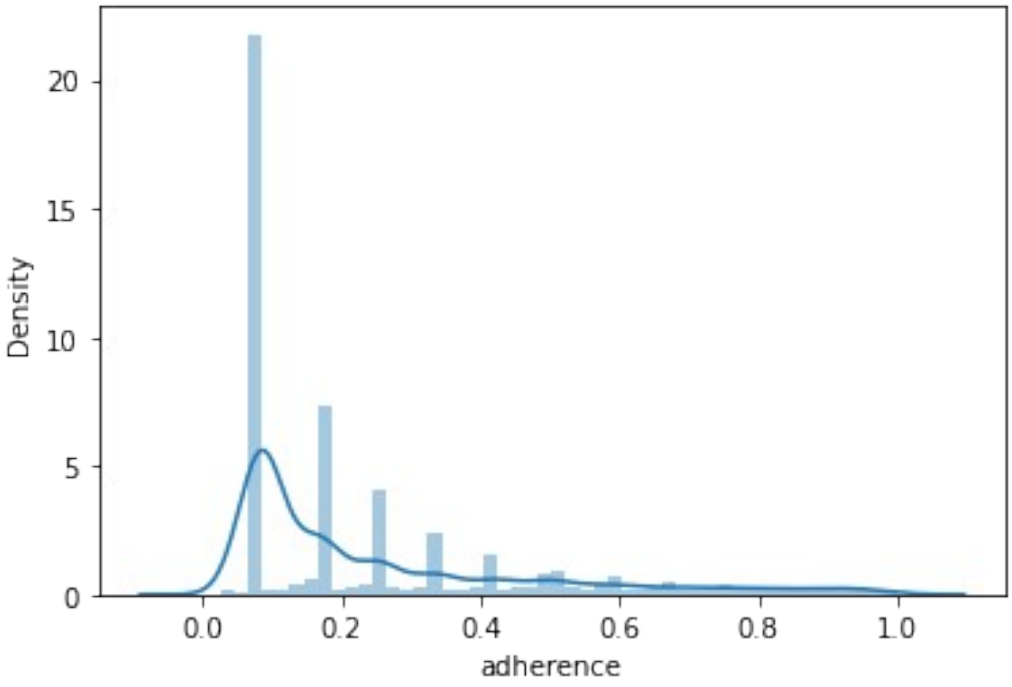

Density
sinusitis

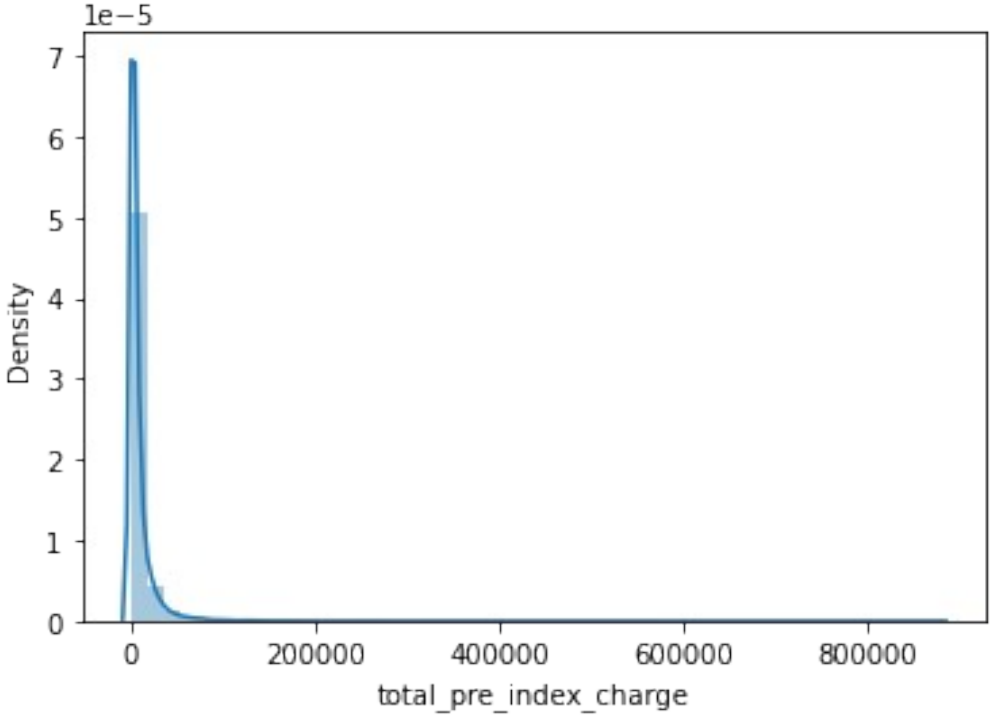

Density
acute_bronchitis

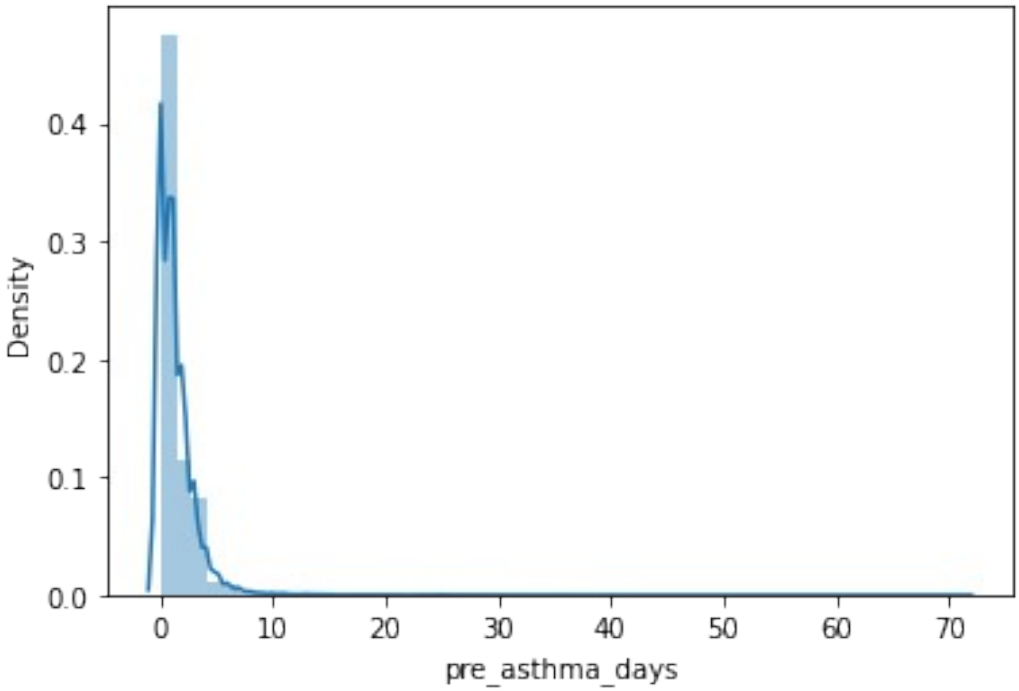

Density
50
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
acute_laryngitis

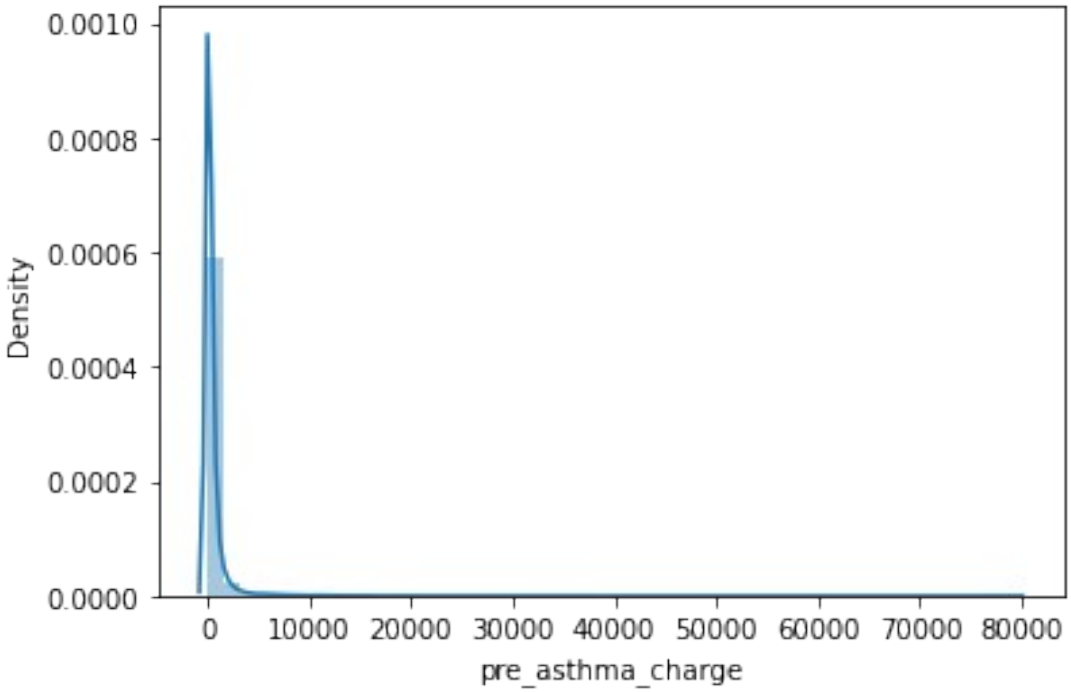

Density
40
35
30
25
20
15
10
5
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
upper_respiratory_infection

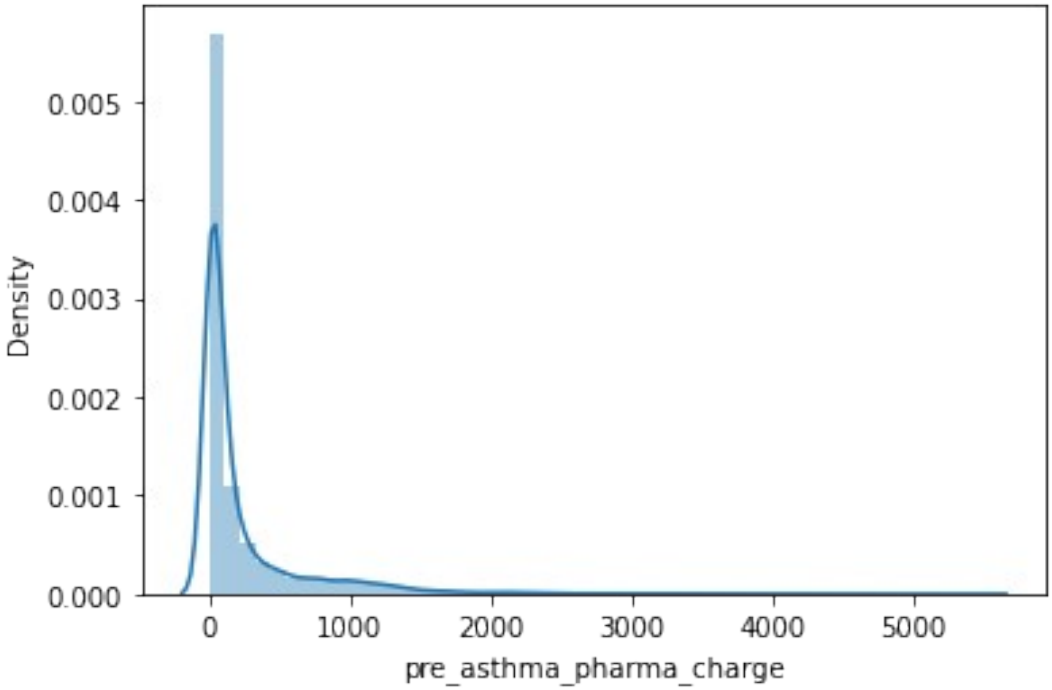

Density
rhinitis
0
1
2
3
4
5
6
7
8
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

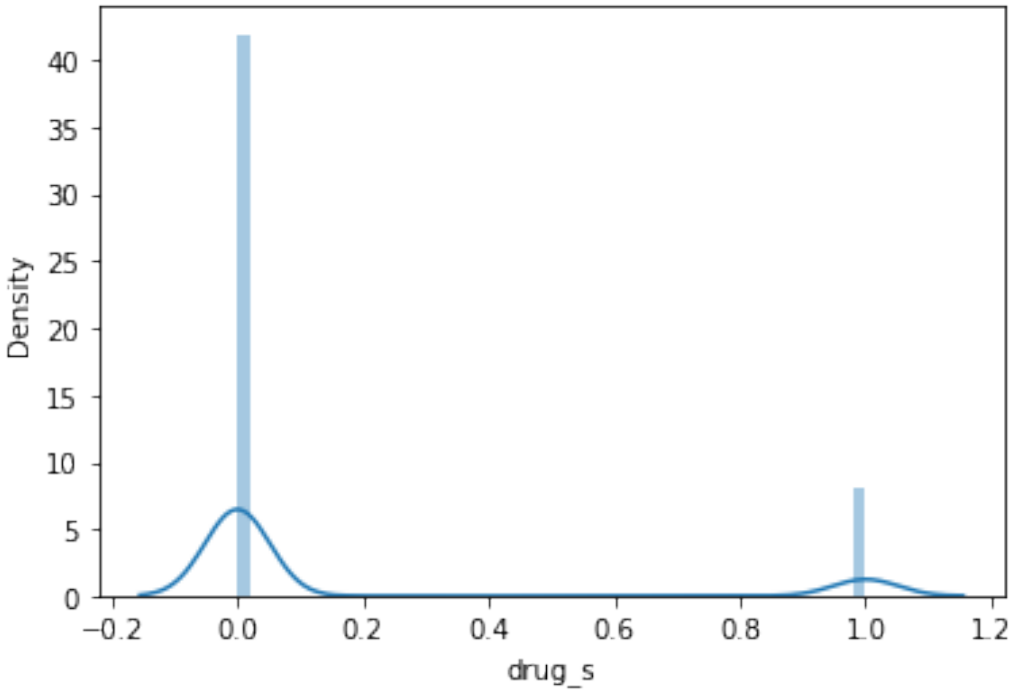

Density
adherence
0
5
10
15
20
0.0
0.2
0.4
0.6
0.8
1.0

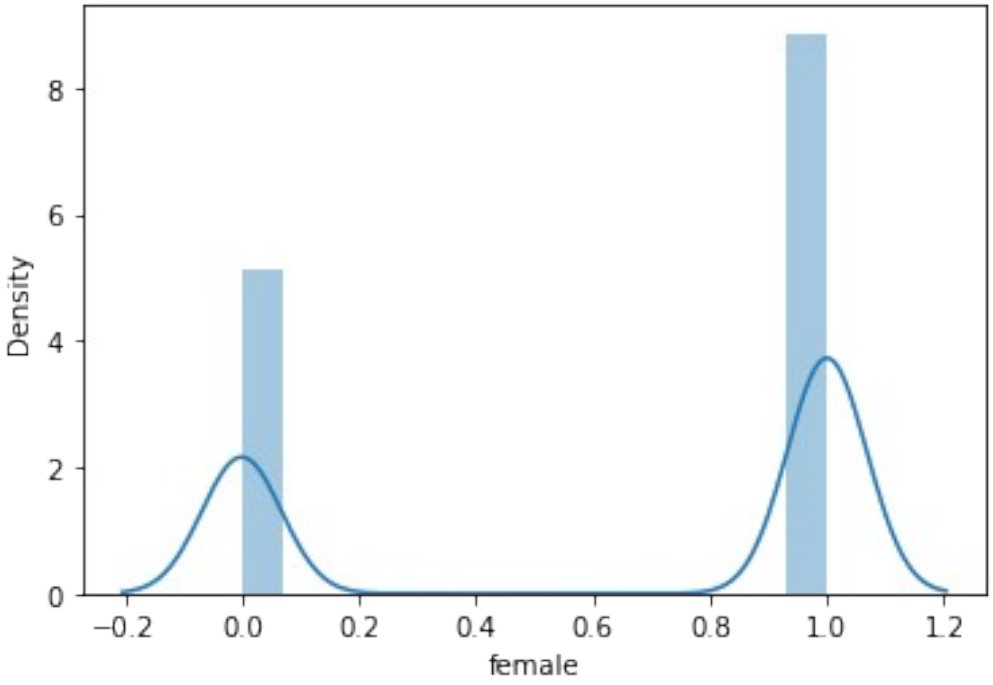

1e−5
7
6
5
4
3
2
1
0
Density
0
200000
400000
600000
800000
total_pre_index_charge

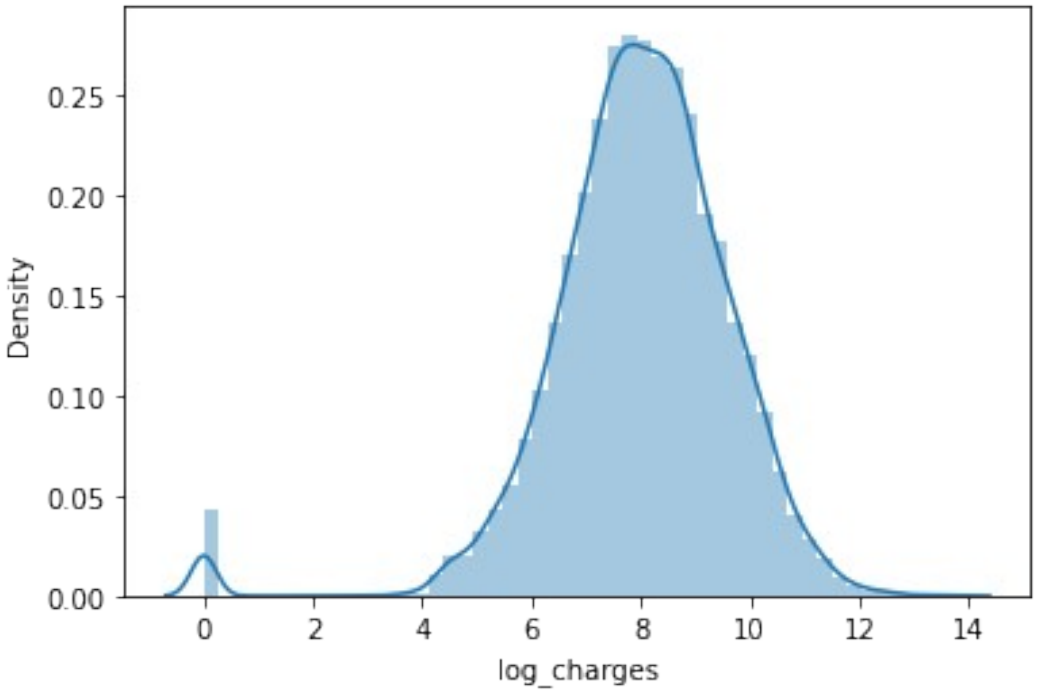

0.4
0.3
0.2
0.1
0.0
Density
0
10
20
30
40
50
60
70
pre_asthma_days

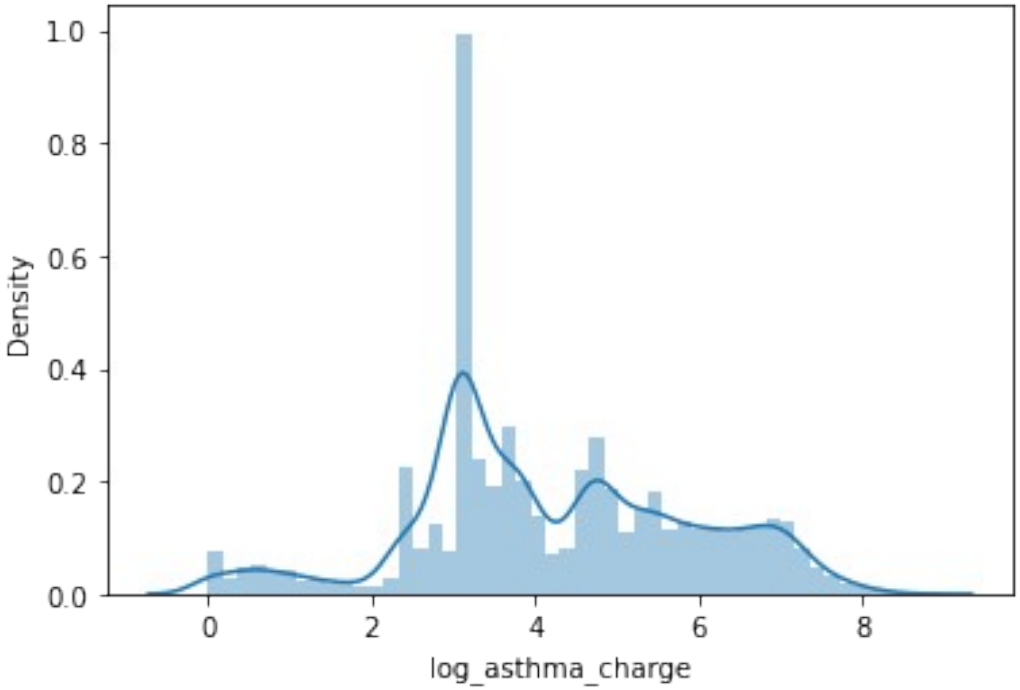

0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000
50000
60000
70000
80000
pre_asthma_charge


0.005
0.004
0.003
0.002
0.001
0.000
Density
0
1000
2000
3000
4000
5000
pre_asthma_pharma_charge


Density
40
35
30
25
20
15
10
5
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
drug_s


Density
8
6
4
2
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
female





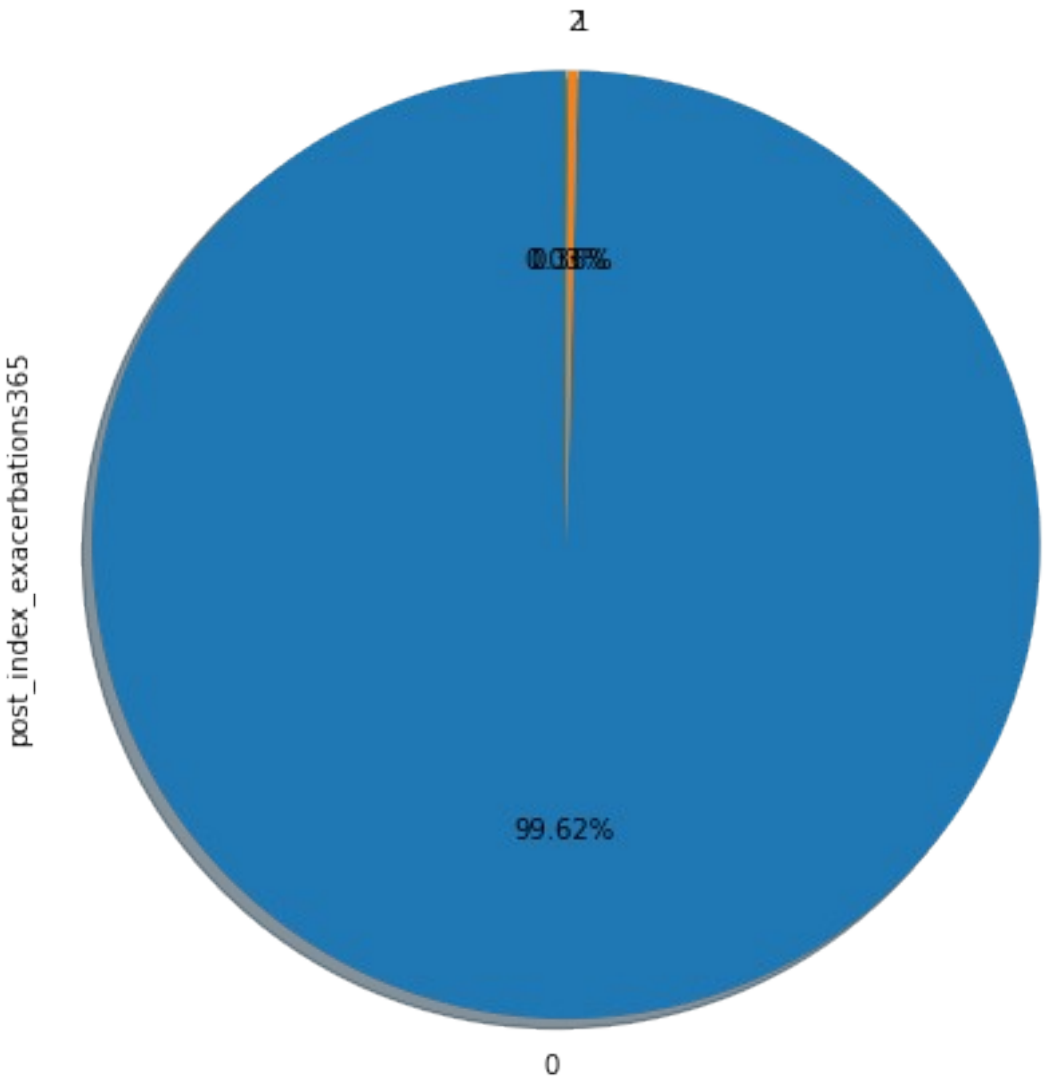

```
df['post_index_exacerbations365'].value_counts()
```

```
0     16124
1      1525
2       333
3       115
4        48
```

```
5        32
6        12
8         9
7         9
9         3
10        2
12        1
13        1
14        1
Name: post_index_exacerbations365, dtype: int64
```

Below, i have created one function to categorized the data into 3 categories i.e 0,1 and 2. so that i can compare my data with pre drug count.

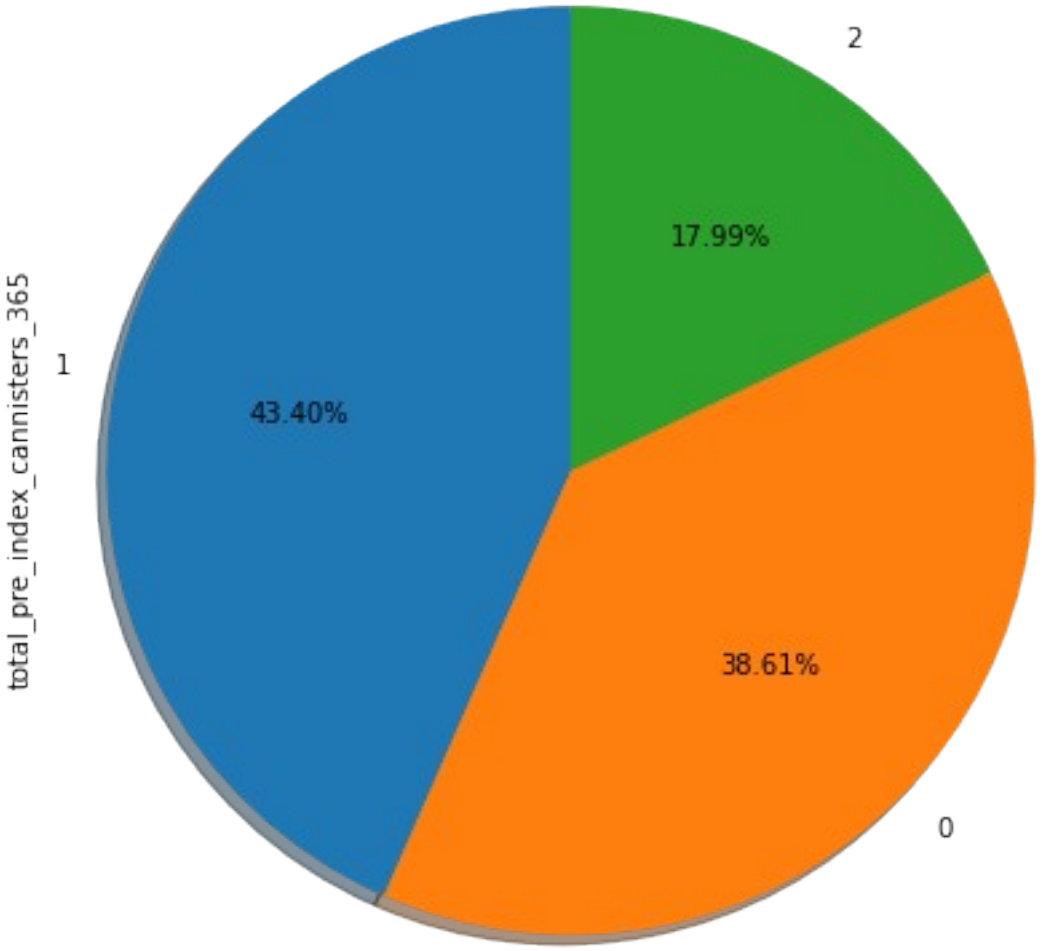
```
def post_data(x):
    if x in list(range(0,5)):
        return 0
    elif x in list(range(5,10)):
        return 1
    elif x in list(range(10,15)):
        return 2


df["post_index_exacerbations365"]=df['post_index_exacerbations365'].ap
ply(post_data)

df['post_index_exacerbations365'].value_counts()

0    18145
1       65
2        5
Name: post_index_exacerbations365, dtype: int64

plt.figure(figsize=(7,7))
df["post_index_exacerbations365"].value_counts().plot.pie(shadow=True,
startangle=90, autopct="%1.2f%%")
plt.axis('equal')
plt.show()
```




```
df['total_pre_index_cannisters_365'].value_counts()
```

```
1    7906
0    7033
2    3276
Name: total_pre_index_cannisters_365, dtype: int64
```

```
plt.figure(figsize=(7,7))
df["total_pre_index_cannisters_365"].value_counts().plot.pie(shadow=Tr
ue, startangle=90, autopct="%1.2f%%")
plt.axis('equal')
plt.show()
```



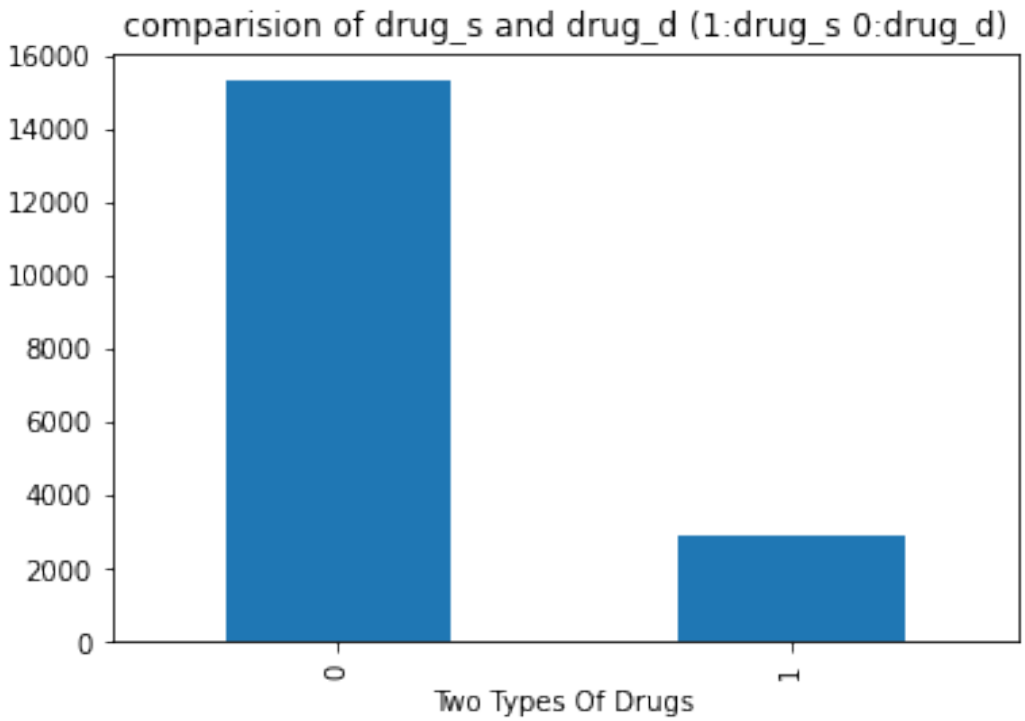

```
df.drop(['patid'],inplace=True,axis=1)
# Here i am dropping id column because i find the relationship between
id and the data i am going to calculate have the least effect on it

df.head()
```

	index_age	previous_asthma_drugs	total_pre_index_cannisters_365	post_index_exacerbations365	pneumonia	sinusitis	acute_bronchitis
0	14	1	1	0	0	0	1
1	21	1	2	0	0	1	1
2	62	1	0				
3	30	1	2				
4	40	1	1				

```
2                           0          0          0                 0

3                           0          0          0                 1

4                           0          0          1                 0


   acute_laryngitis  upper_respiratory_infection  gerd  rhinitis
adherence  \
0                 0                            1     0         0
0.084469
1                 0                            0     0         1
0.084469
2                 0                            0     0         0
0.738420
3                 0                            1     0         0
0.084469
4                 1                            1     0         1
0.506812

   total_pre_index_charge  pre_asthma_days  pre_asthma_charge  \
0             1224.767473                1              314.0
1            20290.534269                0                0.0
2             2964.254175                4              480.0
3             3223.708820                0                0.0
4             1287.254368                2              689.0

   pre_asthma_pharma_charge  drug_s  female  log_charges
log_asthma_charge
0                    218.13       0       0     7.110506
5.385091
1                     44.98       0       0     9.917910
3.806218
2                     99.26       0       0     7.994381
4.597743
3                     59.58       0       1     8.078288
4.087320
4                     29.79       0       0     7.160267
3.394173
```

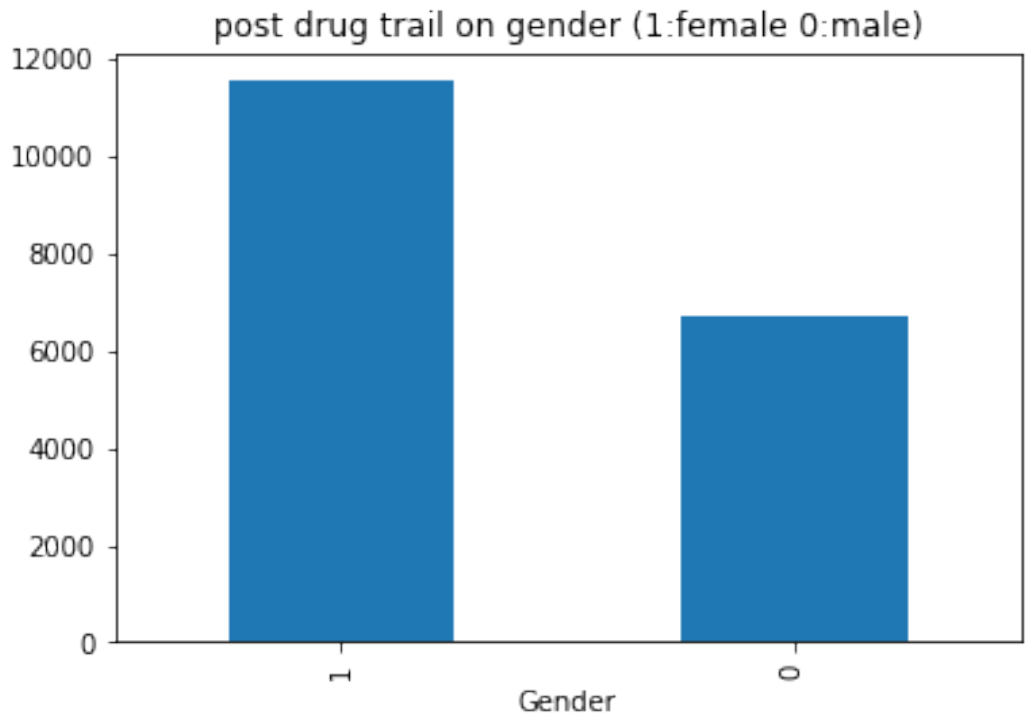
Plotted the comparision between the drug_s and drug_d

```
df["drug_s"].value_counts().plot(kind="bar")
plt.title("comparision of drug_s and drug_d (1:drug_s 0:drug_d)")
plt.xlabel("Two Types Of Drugs")
plt.show()
```



Drug count on female and men in total

```
df["female"].value_counts().plot(kind="bar")
plt.title("post drug trail on gender (1:female 0:male)")
plt.xlabel("Gender")
plt.show()
```




```
x1=df.drop(["total_pre_index_cannisters_365",'post_index_exacerbations
365'],axis=1)
x1.head()
```

```
   index_age  previous_asthma_drugs  pneumonia  sinusitis
acute_bronchitis  \
0         14                      1          0          0
1
1         21                      1          0          1
1
2         62                      1          0          0
0
3         30                      1          0          0
1
4         40                      1          0          1
0

   acute_laryngitis  upper_respiratory_infection  gerd  rhinitis
adherence  \
0                 0                            1     0         0
0.084469
1                 0                            0     0         1
0.084469
2                 0                            0     0         0
0.738420
3                 0                            1     0         0
0.084469
4                 1                            1     0         1
```

```
0.506812

   total_pre_index_charge  pre_asthma_days  pre_asthma_charge  \
0             1224.767473                1              314.0
1            20290.534269                0                0.0
2             2964.254175                4              480.0
3             3223.708820                0                0.0
4             1287.254368                2              689.0

   pre_asthma_pharma_charge  drug_s  female  log_charges
log_asthma_charge
0                    218.13       0       0     7.110506
5.385091
1                     44.98       0       0     9.917910
3.806218
2                     99.26       0       0     7.994381
4.597743
3                     59.58       0       1     8.078288
4.087320
4                     29.79       0       0     7.160267
3.394173
```

```
y1=df["total_pre_index_cannisters_365"]
y1
```

```
0        1
1        2
2        0
3        2
4        1
        ..
18210    1
18211    0
18212    2
18213    1
18214    1
Name: total_pre_index_cannisters_365, Length: 18215, dtype: int64
```

#Splitting data and creating Pre-Drug Model

```
#pre drugs model

from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x1,y1,test_size=0.30,
random_state=1)

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

def mymodel(model):
  model.fit(xtrain,ytrain)
  ypred=model.predict(xtest)
  print(classification_report(ytest,ypred))

mymodel(RandomForestClassifier())
```

	precision	recall	f1-score	support
0	0.67	0.82	0.74	2146
1	0.75	0.70	0.72	2320
2	0.68	0.45	0.54	999
accuracy			0.70	5465
macro avg	0.70	0.65	0.66	5465
weighted avg	0.70	0.70	0.69	5465

```
mymodel(DecisionTreeClassifier())
```

	precision	recall	f1-score	support
0	0.67	0.67	0.67	2146
1	0.67	0.67	0.67	2320
2	0.50	0.50	0.50	999
accuracy			0.64	5465
macro avg	0.61	0.61	0.61	5465
weighted avg	0.64	0.64	0.64	5465

```
from sklearn.ensemble import BaggingClassifier
mymodel(BaggingClassifier())
```

	precision	recall	f1-score	support
0	0.67	0.79	0.73	2146
1	0.72	0.70	0.71	2320
2	0.63	0.45	0.52	999
accuracy			0.69	5465
macro avg	0.67	0.64	0.65	5465
weighted avg	0.69	0.69	0.68	5465

#Post-Drug model

```
x2=df.drop(["total_pre_index_cannisters_365",'post_index_exacerbations
365'],axis=1)
x2.head()
```

```
   index_age  previous_asthma_drugs  pneumonia  sinusitis
acute_bronchitis  \
0         14                      1          0          0
1
1         21                      1          0          1
1
2         62                      1          0          0
0
3         30                      1          0          0
1
4         40                      1          0          1
0

   acute_laryngitis  upper_respiratory_infection  gerd  rhinitis
adherence  \
0                 0                            1     0         0
0.084469
1                 0                            0     0         1
0.084469
2                 0                            0     0         0
0.738420
3                 0                            1     0         0
0.084469
4                 1                            1     0         1
0.506812

   total_pre_index_charge  pre_asthma_days  pre_asthma_charge  \
0             1224.767473                1              314.0
1            20290.534269                0                0.0
2             2964.254175                4              480.0
3             3223.708820                0                0.0
4             1287.254368                2              689.0

   pre_asthma_pharma_charge  drug_s  female  log_charges
log_asthma_charge
0                    218.13       0       0     7.110506
5.385091
1                     44.98       0       0     9.917910
3.806218
2                     99.26       0       0     7.994381
4.597743
3                     59.58       0       1     8.078288
4.087320
4                     29.79       0       0     7.160267
3.394173
```

```
y2=df["post_index_exacerbations365"]
y2
```

```
0        0
1        0
2        0
3        0
4        0
        ..
18210    0
18211    0
18212    0
18213    0
18214    0
Name: post_index_exacerbations365, Length: 18215, dtype: int64
```

Below i have done Random Over Sampling Because the data is highly imbalance i.e it has an unequal number of observations.

```
from sklearn.datasets import make_classification
x2, y2 = make_classification(n_samples=18215)
from imblearn.over_sampling import RandomOverSampler
ros=RandomOverSampler()
x2,y2=ros.fit_resample(x2,y2)
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x2,y2,test_size=0.3,random_
state=0)
```

```
def mymodel1(model):
  model.fit(xtrain,ytrain)
  ypred=model.predict(xtest)
  print(classification_report(ytest,ypred))

mymodel1(RandomForestClassifier())
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      2744
           1       0.97      0.98      0.97      2729

    accuracy                           0.97      5473
   macro avg       0.97      0.97      0.97      5473
weighted avg       0.97      0.97      0.97      5473


mymodel1(DecisionTreeClassifier())

              precision    recall  f1-score   support

           0       0.94      0.95      0.94      2744
           1       0.95      0.94      0.94      2729
```

accuracy			0.94	5473
macro avg	0.94	0.94	0.94	5473
weighted avg	0.94	0.94	0.94	5473

mymodel1(BaggingClassifier())

	precision	recall	f1-score	support
0	0.97	0.97	0.97	2744
1	0.97	0.97	0.97	2729
accuracy			0.97	5473
macro avg	0.97	0.97	0.97	5473
weighted avg	0.97	0.97	0.97	5473