EDA

August 10, 2022

0.1 EDA Performed on the Data

[3]:
```python
# Read the csv dataset
data = pd.read_csv('Healthcare_dataset.csv')

# Drop the ID variable
data = data.drop(["Ptid"], axis=1)

data.head()
```

[3]:
```
  Persistency_Flag  Gender           Race      Ethnicity   Region Age_Bucket  \
0       Persistent    Male      Caucasian  Not Hispanic     West        >75
1   Non-Persistent    Male          Asian  Not Hispanic     West      55-65
2   Non-Persistent  Female  Other/Unknown      Hispanic  Midwest      65-75
3   Non-Persistent  Female      Caucasian  Not Hispanic  Midwest        >75
4   Non-Persistent  Female      Caucasian  Not Hispanic  Midwest        >75

         Ntm_Speciality Ntm_Specialist_Flag      Ntm_Speciality_Bucket  \
0  GENERAL PRACTITIONER              Others  OB/GYN/Others/PCP/Unknown
1  GENERAL PRACTITIONER              Others  OB/GYN/Others/PCP/Unknown
2  GENERAL PRACTITIONER              Others  OB/GYN/Others/PCP/Unknown
3  GENERAL PRACTITIONER              Others  OB/GYN/Others/PCP/Unknown
4  GENERAL PRACTITIONER              Others  OB/GYN/Others/PCP/Unknown

  Gluco_Record_Prior_Ntm  … Risk_Family_History_Of_Osteoporosis  \
0                      N  …                                   N
1                      N  …                                   N
2                      N  …                                   N
3                      N  …                                   N
4                      Y  …                                   N

   Risk_Low_Calcium_Intake Risk_Vitamin_D_Insufficiency  \
0                        N                            N
1                        N                            N
2                        Y                            N
3                        N                            N
4                        N                            N
```

```
  Risk_Poor_Health_Frailty Risk_Excessive_Thinness  \
0                        N                       N
1                        N                       N
2                        N                       N
3                        N                       N
4                        N                       N

  Risk_Hysterectomy_Oophorectomy Risk_Estrogen_Deficiency Risk_Immobilization  \
0                              N                        N                   N
1                              N                        N                   N
2                              N                        N                   N
3                              N                        N                   N
4                              N                        N                   N

  Risk_Recurring_Falls Count_Of_Risks
0                    N              0
1                    N              0
2                    N              2
3                    N              1
4                    N              1

[5 rows x 68 columns]
```

1 Data Cleaning

[4]:
```python
# Total number of missing values
data.isnull().sum().sum()
```

[4]:
```
0
```

Note: The data does not contain any missing values.

[5]:
```python
# Dimension of the dataset
data.shape
```

[5]:
```
(3424, 68)
```

[6]:
```python
# Standardize column names

columns = list(data.columns)

for item in columns:
    special_characters = "!@#$%^&*()-+?=,<>/"

    for character in special_characters:
        if character in item:
            print(f"Feature: {item}, is not well formatted")
```

Feature: Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx, is not well formatted

[7]:
```python
# Remove the "," from this feature name

data["Comorb_Encntr_For_General_Exam_W_O_Complaint_Susp_Or_Reprtd_Dx"] =␣
↪data["Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx"]
data = data.
↪drop(["Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx"],␣
↪axis=1)
data.shape
```

[7]:
```
(3424, 68)
```

[8]:
```python
# Numeric Columns
numeric_col = list(data._get_numeric_data().columns)
numeric_col
```

[8]:
```
['Dexa_Freq_During_Rx', 'Count_Of_Risks']
```

1.0.1 Remove Outliers

[9]:
```python
data['Dexa_Freq_During_Rx'].value_counts()
```

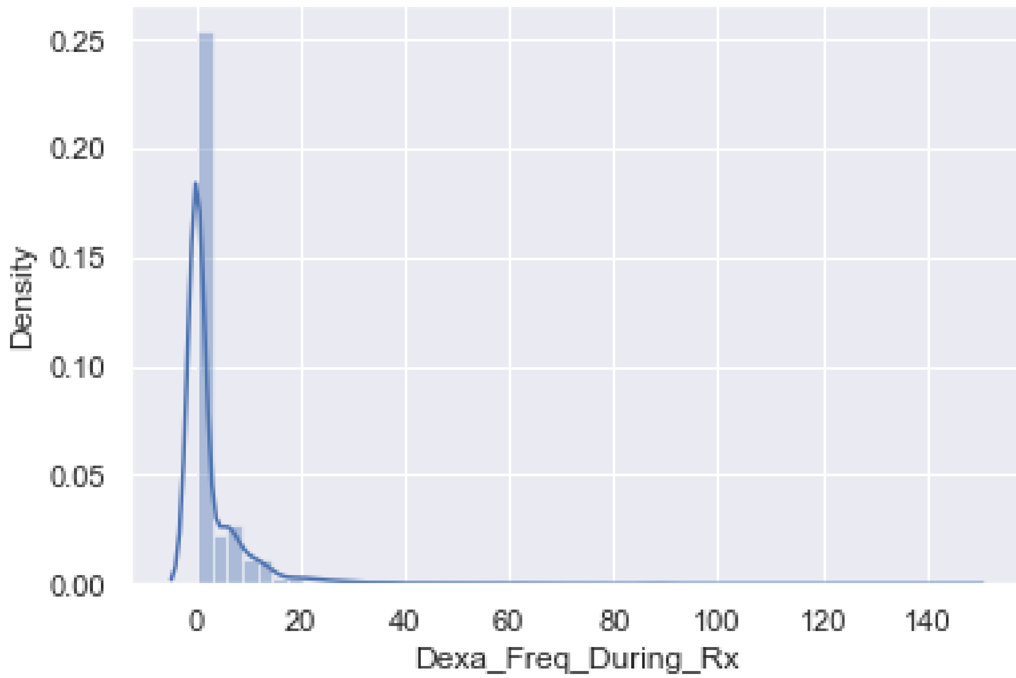
[9]:
```
0      2488
5       114
6       107
7        93
8        71
4        68
10       55
12       52
3        46
14       38
9        32
11       30
1        24
2        24
13       19
20       15
16       14
18       14
22       13
26       10
24       10
15        9
30        7
17        7
```

```
28        7
21        7
36        5
19        3
42        3
32        3
34        3
52        2
48        2
58        2
25        2
39        2
88        2
54        1
146       1
50        1
35        1
44        1
108       1
72        1
40        1
68        1
45        1
38        1
69        1
118       1
66        1
110       1
33        1
23        1
27        1
81        1
37        1
29        1
Name: Dexa_Freq_During_Rx, dtype: int64
```

[10]:

```python
# Spot Outliers for Dexa_Freq_During_Rx
sns.distplot(data['Dexa_Freq_During_Rx'])
```

[10]: <AxesSubplot:xlabel='Dexa_Freq_During_Rx', ylabel='Density'>



Note: The outliers in the data are skewed towards the right, to fix this we remove the 97th percentile.

[11]:
```
# To remove the 99th percentile
q = data['Dexa_Freq_During_Rx'].quantile(0.97)
data_1 = data[data['Dexa_Freq_During_Rx']<q]
data_1['Dexa_Freq_During_Rx'].describe()
```

[11]:
```
count    3308.000000
mean        1.895707
std         3.835797
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max        19.000000
Name: Dexa_Freq_During_Rx, dtype: float64
```

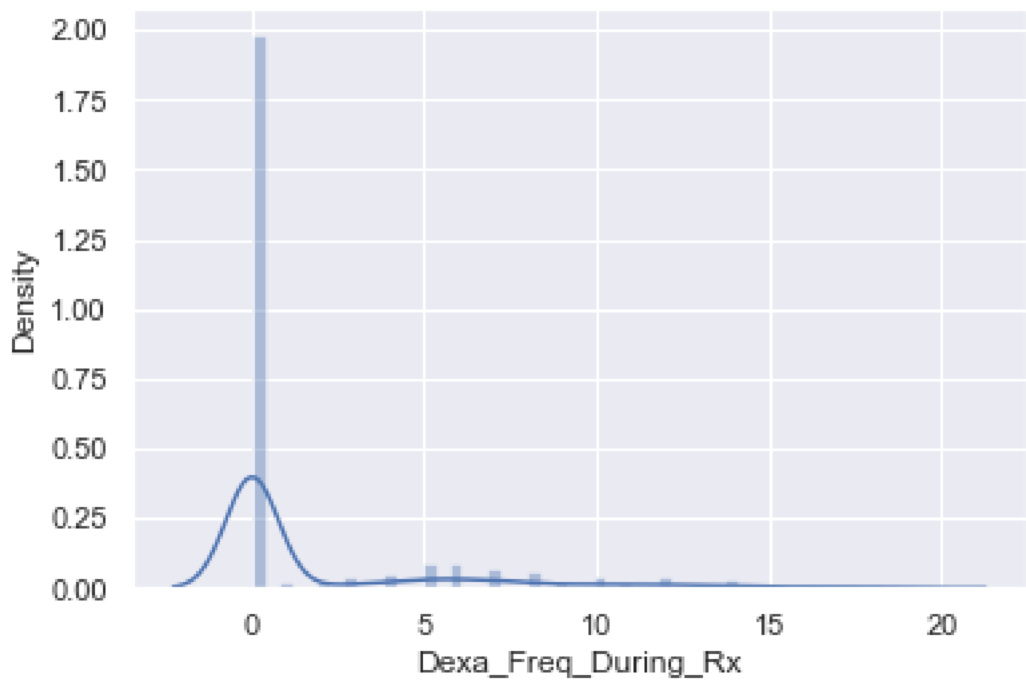
[12]:
```
data_1['Dexa_Freq_During_Rx'].value_counts()
```

[12]:
```
0     2488
5      114
6      107
7       93
8       71
4       68
```

```
10    55
12    52
3     46
14    38
9     32
11    30
1     24
2     24
13    19
18    14
16    14
15     9
17     7
19     3
Name: Dexa_Freq_During_Rx, dtype: int64
```

[13]:
```
sns.distplot(data_1['Dexa_Freq_During_Rx'])
```

[13]: <AxesSubplot:xlabel='Dexa_Freq_During_Rx', ylabel='Density'>



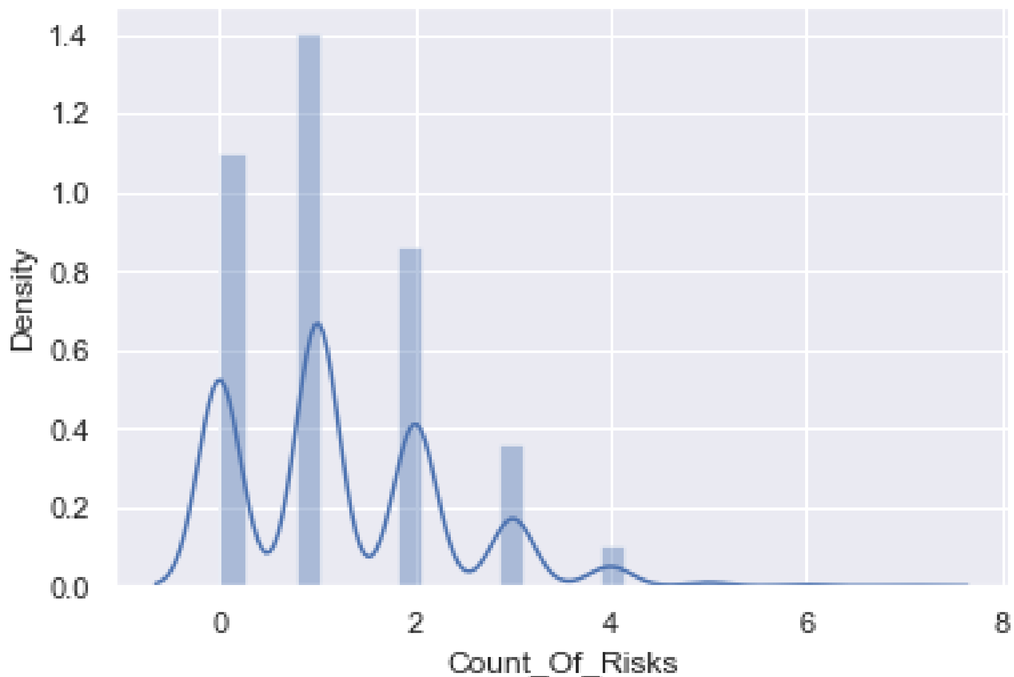
Note: The variable now contains less outliers.

[14]:
```
data_1['Count_Of_Risks'].value_counts()
```

[14]:
```
1    1203
0     943
2     742
3     308
4      89
5      15
6       6
7       2
Name: Count_Of_Risks, dtype: int64
```

[15]:
```
# Spot Outliers for Count_Of_Risks
sns.distplot(data_1['Count_Of_Risks'])
```

[15]:
```
<AxesSubplot:xlabel='Count_Of_Risks', ylabel='Density'>
```



Note: In this variable, the outliers are also skewed towards the right. To fix this we remove the 99th percentile.

[16]:
```
# To remove the 99th percentile
q = data_1['Count_Of_Risks'].quantile(0.99)
data_2 = data_1[data_1['Count_Of_Risks']<q]
data_2.describe()
```

[16]:
```
       Dexa_Freq_During_Rx  Count_Of_Risks
count          3196.000000     3196.000000
```

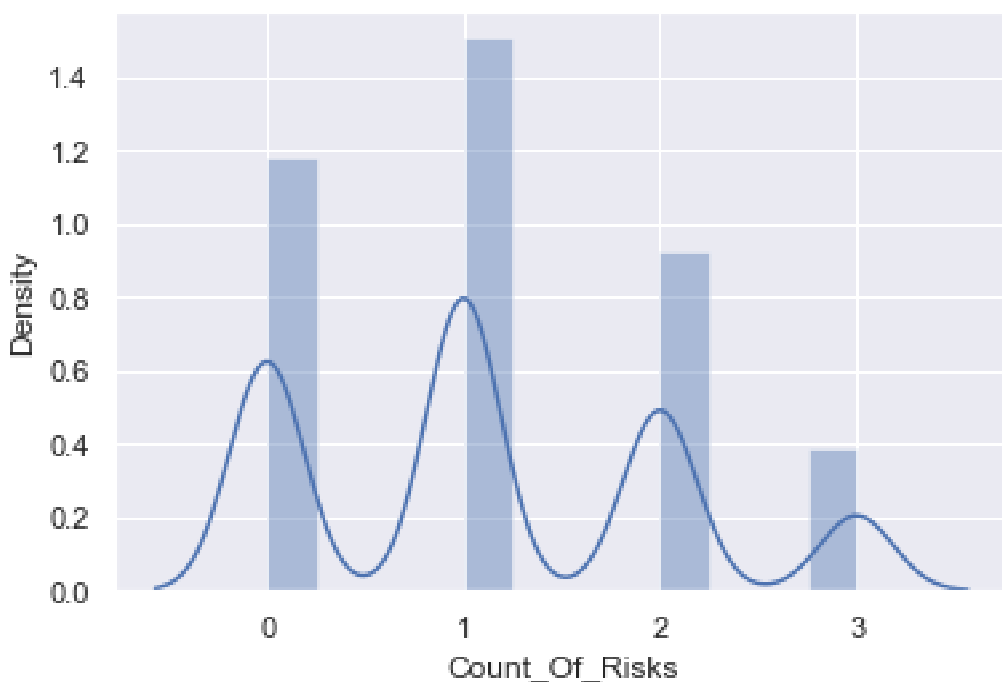
```
mean              1.875782        1.129850
std               3.834596        0.946638
min               0.000000        0.000000
25%               0.000000        0.000000
50%               0.000000        1.000000
75%               0.000000        2.000000
max              19.000000        3.000000
```

[17]:
```
data_2['Count_Of_Risks'].value_counts()
```

[17]:
```
1    1203
0     943
2     742
3     308
Name: Count_Of_Risks, dtype: int64
```

[18]:
```
sns.distplot(data_2['Count_Of_Risks'])
```

[18]:
```
<AxesSubplot:xlabel='Count_Of_Risks', ylabel='Density'>
```



This variable no longer contains outliers.

[19]:
```
# Data without outliers
data = data_2
```

[20]:
```python
# Data containing only categorical variables
categoric_data = data.drop(numeric_col, axis=1)
```

[21]:
```python
# Categorical Columns
cat_columns = list(categoric_data.columns)
```

1.0.2 Encode Target Variable

[23]:
```python
# The target variable
data["Persistency_Flag"].unique()
```

[23]:
```
array(['Persistent', 'Non-Persistent'], dtype=object)
```

[24]:
```python
# Encode the Target variable

# data["Persistency_Flag"] = data["Persistency_Flag"].map({"Non-Persistent":0,␣
↪"Persistent":1})

from sklearn.preprocessing import LabelEncoder

lb_make = LabelEncoder()
data["Persistency_Flag"] = lb_make.fit_transform(data["Persistency_Flag"])

# data["Persistency_Flag"].head()

data["Persistency_Flag"].unique()
```

```
<ipython-input-24-f18f0a11ba41>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data["Persistency_Flag"] = lb_make.fit_transform(data["Persistency_Flag"])
```

[24]:
```
array([1, 0])
```

Note: 1 = Persistent, 0 = Non-Persistent

[25]:
```python
data["Persistency_Flag"].unique()
```

[25]:
```
array([1, 0])
```

1.0.3 Balance Dataset

[26]:
```python
data["Persistency_Flag"].value_counts()
```

[26]:
```
0    2070
1    1206
Name: Persistency_Flag, dtype: int64
```

Note: The Non-Persistent observations are almost double the Persistent observations.

Next step is to balance the data using either the Random-Undersampling or Random_Oversampling method.

[27]:
```
# Oversampling

# import library
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler(random_state=42)

# fit predictor and target variablex_ros, y_ros = ros.fit_resample(x, y)
x_temp, y_temp = ros.fit_resample(data.drop(["Persistency_Flag"], axis=1),␣
↪data["Persistency_Flag"])
```

[28]:
```
data = pd.concat([x_temp, y_temp], axis=1)
```

[29]:
```
data["Persistency_Flag"].value_counts()
```

[29]:
```
1    2070
0    2070
Name: Persistency_Flag, dtype: int64
```

The data is now balanced, with 2070 observations of both classes each.