Data Science Health Care Week 8

Group Member (Solo):

- *Name: Halit Ayberk Demir*
- *E-Mail: h.ayberk.demir.34@gmail.com*
- *University/Company: Middle East Technical University/Data Analyst at Mega Pharma*
- *Specialization: Data Science*

Problem Description

ABC is a pharmaceutical company and desires to understand the persistency of the drug per physician description. To solve this issue, ABC company reached an analytics company to automate this process of identification.

Data Understanding

Healthcare dataset is xlsx (Excel Workbook) which has 2 worksheets, at worksheet 1, 29 Features are explained individually. These features are divided into 6 buckets, this bucket and regarding features are:

1 - Target Variable

- Patient ID

2 - Demographics

- Age
- Race
- Region
- Ethnicity
- Gender
- IDN Indicator

3 - Provider Attributes

- NTM - Physician Specialty

4 - Clinical Factors

- NTM - T-Score
- Change in T Score

- NTM - Risk Segment
- Change in Risk Segment
- NTM - Multiple Risk Factors
- NTM - Dexa Scan Frequency
- NTM - Dexa Scan Recency
- Dexa During Therapy
- NTM - Fragility Fracture Recency
- Fragility Fracture During Therapy
- NTM - Glucocorticoid Recency
- Glucocorticoid Usage During Therapy

5 - Disease/Treatment Factor

- NTM - Injectable Experience
- NTM - Risk Factors
- NTM - Comorbidity
- NTM - Concomitancy
- Adherence

This features are in the columns of the dataset. Our actual dataset in worksheet number two. The actual dataset contains 3424 columns and 69 columns. When I investigated the dataset at first glance, I could see that most of the columns are either binary or categorical features.

```
RangeIndex: 3424 entries, 0 to 3423
Data columns (total 69 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Ptid                                                    3424 non-null   object
 1   Persistency_Flag                                        3424 non-null   object
 2   Gender                                                  3424 non-null   object
 3   Race                                                    3424 non-null   object
 4   Ethnicity                                               3424 non-null   object
 5   Region                                                  3424 non-null   object
 6   Age_Bucket                                              3424 non-null   object
 7   Ntm_Speciality                                          3424 non-null   object
 8   Ntm_Specialist_Flag                                     3424 non-null   object
 9   Ntm_Speciality_Bucket                                   3424 non-null   object
 10  Gluco_Record_Prior_Ntm                                  3424 non-null   object
 11  Gluco_Record_During_Rx                                  3424 non-null   object
 12  Dexa_Freq_During_Rx                                     3424 non-null   int64
 13  Dexa_During_Rx                                          3424 non-null   object
 14  Frag_Frac_Prior_Ntm                                     3424 non-null   object
 15  Frag_Frac_During_Rx                                     3424 non-null   object
 16  Risk_Segment_Prior_Ntm                                  3424 non-null   object
 17  Tscore_Bucket_Prior_Ntm                                 3424 non-null   object
 18  Risk_Segment_During_Rx                                  3424 non-null   object
 19  Tscore_Bucket_During_Rx                                 3424 non-null   object
...
 67  Risk_Recurring_Falls                                    3424 non-null   object
 68  Count_Of_Risks                                          3424 non-null   int64
dtypes: int64(2), object(67)
memory usage: 1.8+ MB
```

When I First read the dataset its memory usage is around 1.8 Mb, because pandas read the categorical values as objects. I wrote a function to turn the columns which has less than 50 unique values to categorical datatype:

```
df.loc[:, df.nunique() < 50] = df.loc[:, df.nunique() < 50].astype('category')
df.info()
```

```
 2   Gender                                                    3424 non-null   category
 3   Race                                                      3424 non-null   category
 4   Ethnicity                                                 3424 non-null   category
 5   Region                                                    3424 non-null   category
 6   Age_Bucket                                                3424 non-null   category
 7   Ntm_Speciality                                            3424 non-null   category
 8   Ntm_Specialist_Flag                                       3424 non-null   category
 9   Ntm_Speciality_Bucket                                     3424 non-null   category
 10  Gluco_Record_Prior_Ntm                                    3424 non-null   category
 11  Gluco_Record_During_Rx                                    3424 non-null   category
 12  Dexa_Freq_During_Rx                                       3424 non-null   int64
 13  Dexa_During_Rx                                            3424 non-null   category
 14  Frag_Frac_Prior_Ntm                                       3424 non-null   category
 15  Frag_Frac_During_Rx                                       3424 non-null   category
 16  Risk_Segment_Prior_Ntm                                    3424 non-null   category
 17  Tscore_Bucket_Prior_Ntm                                   3424 non-null   category
 18  Risk_Segment_During_Rx                                    3424 non-null   category
 19  Tscore_Bucket_During_Rx                                   3424 non-null   category
...
 67  Risk_Recurring_Falls                                      3424 non-null   category
 68  Count_Of_Risks                                            3424 non-null   category
dtypes: category(67), int64(1), object(1)
memory usage: 287.6+ KB
```

It's memory usage dropped to 287 Kb's as you can see. Memory usage dropped by 83.3 %. That proves, most of the columns in this data are either binary or consist of categorical data. In order to find the binary columns, I wrote a function:

```
# Binary Values

for name, column in df.items():
    if column.nunique() == 2:
        print(name)
```

And found out that 59 of the 69 columns are binary Data.

Same logic for finding the numeric values.

```
# Numeric Values

df.select_dtypes("int64").columns.to_list()
```

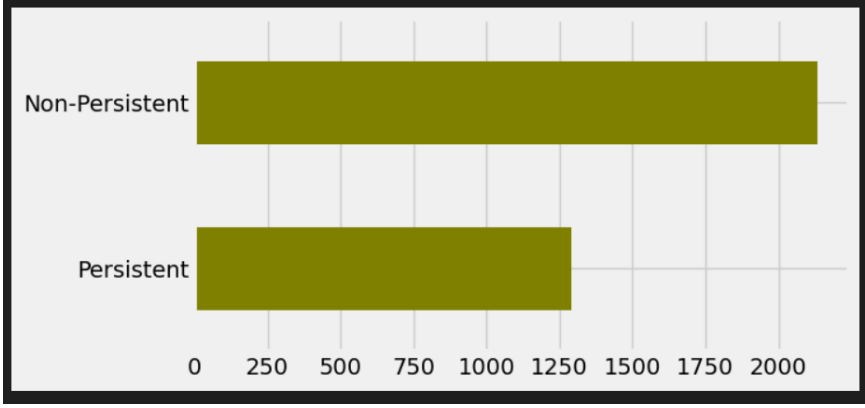
```
['Dexa_Freq_During_Rx', 'Count_Of_Risks']
```

It is clear that we have only two numerical data in our whole dataset.

Investigating Important Columns

Persistency Flag

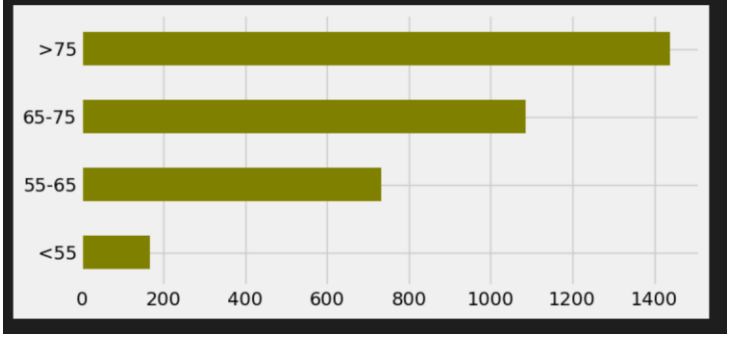
```
Value Counts for the Persistency_Flag
-------------------------------------------------
Non-Persistent    2135
Persistent        1289
Name: Persistency_Flag, dtype: int64
-------------------------------------------------
Column Desctiption for the Persistency_Flag
-------------------------------------------------
count               3424
unique                 2
top       Non-Persistent
freq                2135
Name: Persistency_Flag, dtype: object
-------------------------------------------------
```



Persistency flag is the target variable, we have 2135 Non persistent and 1289 persistent patients. It is clear that non persistent patients are more than persistent patients; thus, we have imbalanced target data.

Age

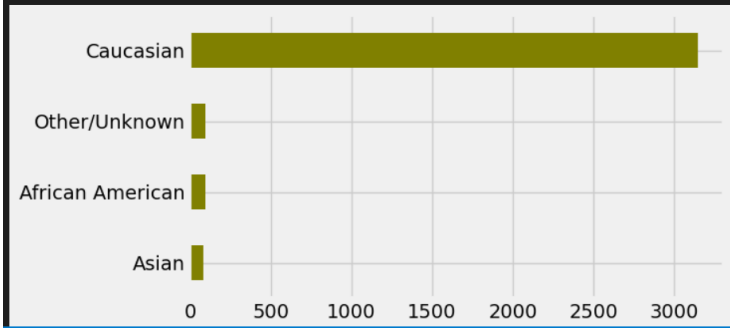
```
Value Counts for the Age_Bucket
--------------------------------------------------
>75      1439
65-75    1086
55-65     733
<55       166
Name: Age_Bucket, dtype: int64
--------------------------------------------------
Column Desctiption for the Age_Bucket
--------------------------------------------------
count     3424
unique       4
top         >75
freq      1439
Name: Age_Bucket, dtype: object
--------------------------------------------------
```



Patients over 75 years are dominating in this column and middle aged (<55) patients are the least common in this dataset.

Race

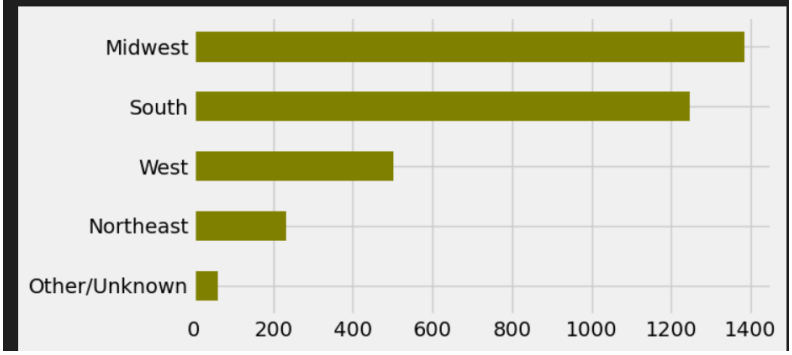
```
--------------------------------------------------
Caucasian           3148
Other/Unknown         97
African American      95
Asian                 84
Name: Race, dtype: int64
--------------------------------------------------
Column Desctiption for the Race
--------------------------------------------------
count          3424
unique            4
top       Caucasian
freq           3148
Name: Race, dtype: object
--------------------------------------------------
```



We have a lot of Caucasian patients. And notice that we have some ***unknown*** values in this column.

Region

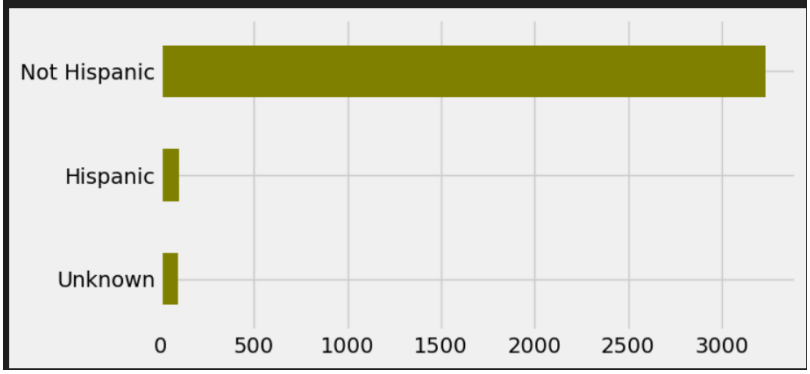
```
Midwest          1383
South            1247
West              502
Northeast         232
Other/Unknown      60
Name: Region, dtype: int64
--------------------------------------------------
Column Desctiption for the Region
--------------------------------------------------
count        3424
unique          5
top       Midwest
freq         1383
Name: Region, dtype: object
--------------------------------------------------
```



We have 60 unknown values!

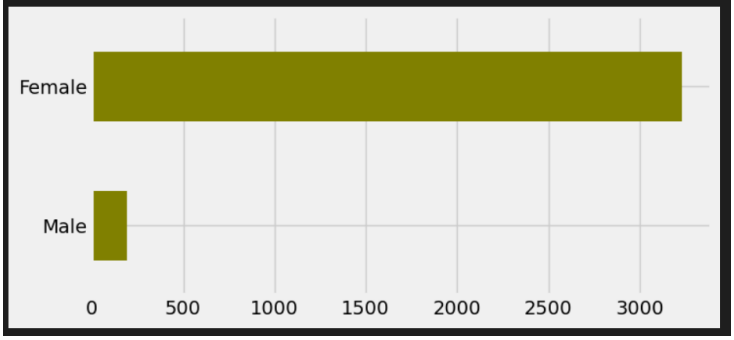
Ethnicity

```
Not Hispanic    3235
Hispanic          98
Unknown           91
Name: Ethnicity, dtype: int64
--------------------------------------------------
Column Desctiption for the Ethnicity
--------------------------------------------------
count             3424
unique               3
top       Not Hispanic
freq              3235
Name: Ethnicity, dtype: object
--------------------------------------------------
```



Gender

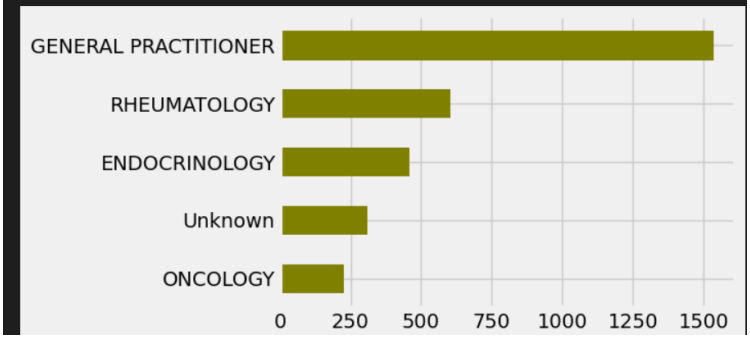
```
--------------------------------------------------
Female    3230
Male       194
Name: Gender, dtype: int64
--------------------------------------------------
Column Desctiption for the Gender
--------------------------------------------------
count       3424
unique         2
top       Female
freq        3230
Name: Gender, dtype: object
--------------------------------------------------
```



It is clear that this disease is much more common at females than males.

NTM Specialty

```
GENERAL PRACTITIONER    1535
RHEUMATOLOGY             604
ENDOCRINOLOGY            458
Unknown                  310
ONCOLOGY                 225
Name: Ntm_Speciality, dtype: int64
--------------------------------------------------
Column Desctiption for the Ntm_Speciality
--------------------------------------------------
count                   3424
unique                    36
top       GENERAL PRACTITIONER
freq                    1535
Name: Ntm_Speciality, dtype: object
--------------------------------------------------
```



NA Values

At first glance it seems like our dataset does not have NaN values, but we can see that there are some values categorized as Unknown and Other/Unkown values. This is a problem for our machine learning model and therefore we have to deal with these values. In order to check out the columns which has these values and the percentage of them:

```python
def detect_none(x):
    if x in ["Unknown", "Other/Unknown"]:
        return None
    else:
        return x

df = df.applymap(detect_none)
```

```
none_columns = df.isna().sum()

percent_none = none_columns[none_columns > 0] / len(df) * 100

percent_none
```

```
Race                        2.832944
Ethnicity                   2.657710
Region                      1.752336
Ntm_Speciality              9.053738
Risk_Segment_During_Rx     43.720794
Tscore_Bucket_During_Rx    43.720794
Change_T_Score             43.720794
Change_Risk_Segment        65.099299
dtype: float64
```

Above you can see the percentages of NaN values. We have 8 columns that have NaN values, 4 of them have above 40% NaN values. For the columns which have more than 40 percent NaN values, we should get rid of those columns because any replacement of those rows will result in an error in our model. For the other NaN values one method could be filling them with the most frequent value in the desired column can be a solution. If the percentage is low enough, we can even delete those rows completely.

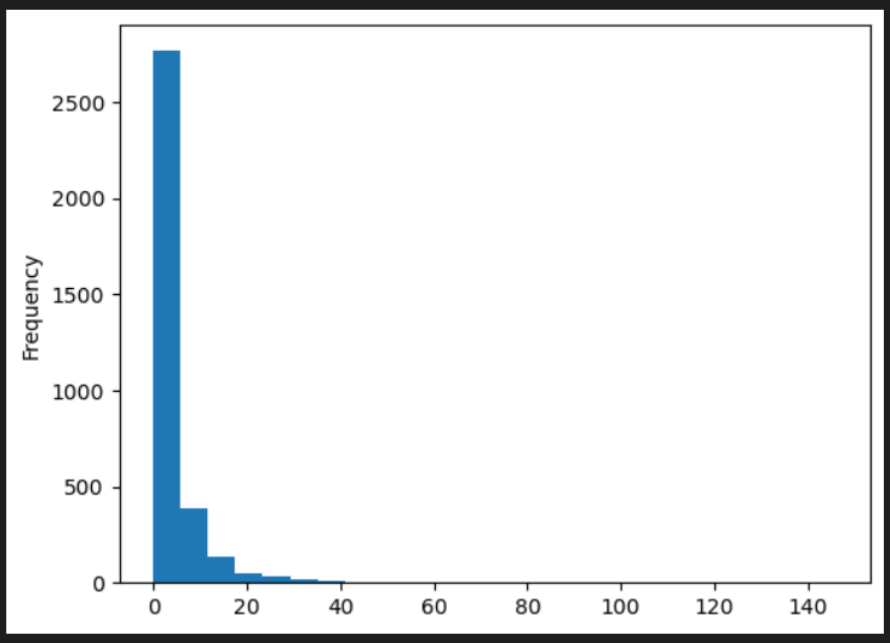
Numerical Values

As I mentioned above, we have 2 numeric columns, and their distributions are:

Dexa Frequency During Rx:

```
df["Dexa_Freq_During_Rx"].plot.hist(bins=25)
plt.show()
```

✓ 1.2s



It’s skewness and kurtosis are:

```
df["Dexa_Freq_During_Rx"].skew()
```

[13] ✓ 0.0s

```
6.8087302112992285
```

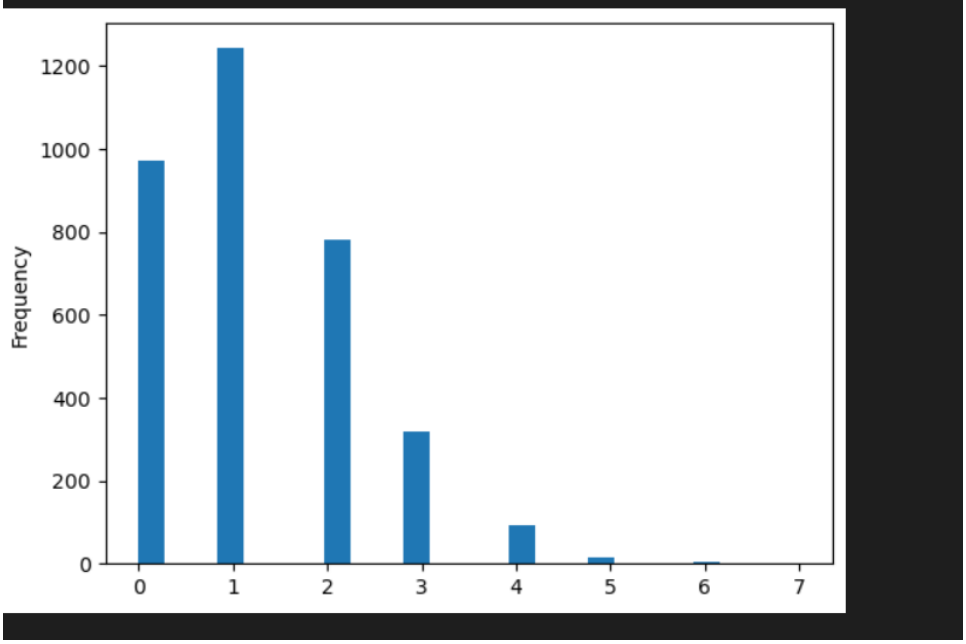
```
df["Dexa_Freq_During_Rx"].kurtosis()
```

[14] ✓ 0.0s

```
74.75837754795428
```

Count Of Risks:

```
df["Count_Of_Risks"].plot.hist(bins=25)
plt.show()
```
✓ 0.1s



Its skewness and kurtosis are:

```
df["Count_Of_Risks"].skew()
```
✓ 0.0s

```
0.8797905232898707
```

```
df["Count_Of_Risks"].kurtosis()
```
✓ 0.0s

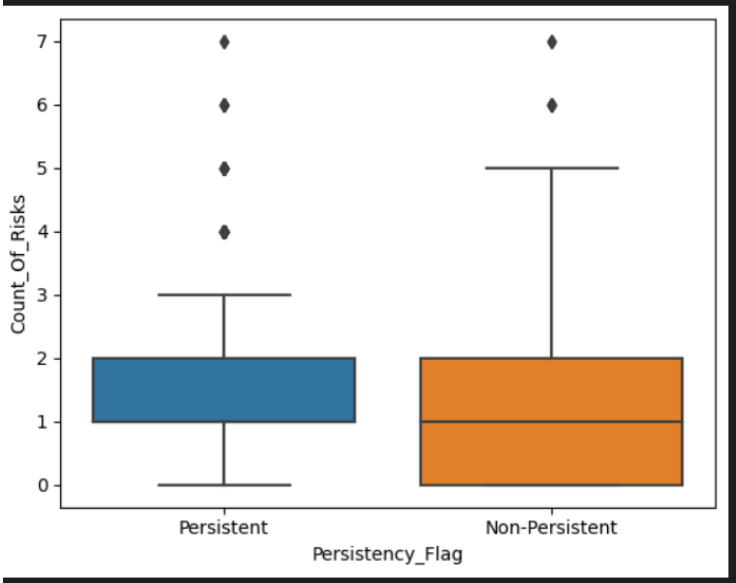
```
0.9004859968892842
```

Outliers

For outliers we can investigate the box plots.

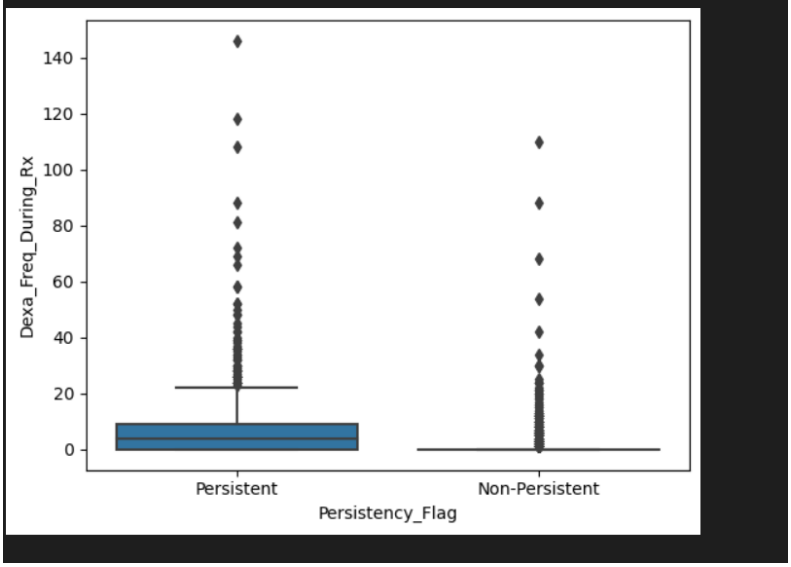
Count of Risks:

```
sns.boxplot(data=df, x="Persistency_Flag", y="Count_Of_Risks")
plt.show()
```



Dexa Frequency:

```
sns.boxplot(data=df, x="Persistency_Flag", y="Dexa_Freq_During_R
plt.show()
```



It is clear that we have outliers in our numerical data we can use interquartile ranges to find these outliers.

```
def find_outliers_IQR(df):
    q1=df.quantile(0.25)
    q3=df.quantile(0.75)
    IQR=q3-q1
    outliers = df[((df<(q1-2*IQR)) | (df>(q3+2*IQR)))]
    return outliers
```

```
find_outliers_IQR(df["Count_Of_Risks"]).count()
```

```
2
```

```
find_outliers_IQR(df["Dexa_Freq_During_Rx"]).count()
```

```
357
```

We have 2 outliers in the count of risks column and 357 outliers in the dexa frequency column. These outliers can be problematic for the machine learning model so getting rid of these outlier values can be the solution to this problem.

Github Repo: https://github.com/demirayberk/Data-Science-HealthCare