Week 9: Deliverables

Team member's details:

Group Name - Chandni

Name – Chandni

Email – cpatel521351@hotmail.com

Country – England

College/Company – NA

Specialization - Data Science

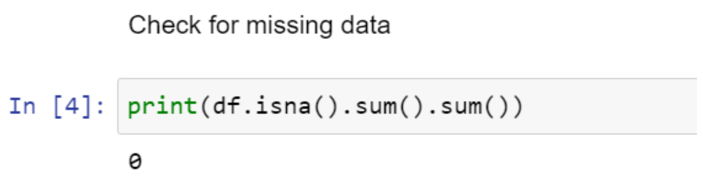
Github - https://github.com/c5678/healthcare/tree/main/3

Problem description

- ABC is a pharma company who wants to understand the persistency of drug as per the physician prescription for the patient. To solve this problem ABC pharma company approached us to automate the process of identification for the drug. This week we are cleaning the how dataset so that it is ready for analysis stage.

Null values

– Firstly we checked if the dataset had any null values which had to be dealt with. No null values were found.

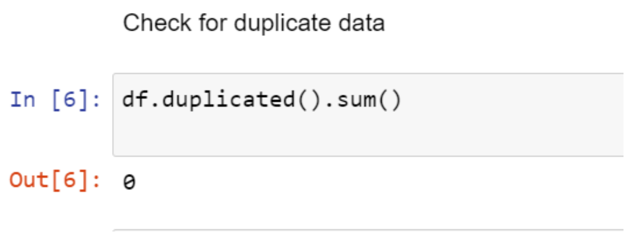


Check for missing data

```
In [4]: print(df.isna().sum().sum())
        0
```

Duplicate values

– No duplicate values were found in the dataset.



Check for duplicate data

```
In [6]: df.duplicated().sum()

Out[6]: 0
```

Removing unwanted data – Firstly Patient ID field was removed because it is of no use for analysis or machine learning model.

Remove Paient ID field which isn't needed

```
In [4]: df.drop(columns= ['Ptid'],inplace= True)
```

We also removed rows of data which had 1490+ “unknown” value to make sure the machine learning model will be as accurate as possible.

1490+ Unknown values were found in 'Risk_Segment_During_Rx', 'Tscore_Bucket_During_Rx', 'Change_T_Score' and 'Change_Risk_Segment'.

Remove 'Risk_Segment_During_Rx','Tscore_Bucket_During_Rx','Change_T_Score','Change_Risk_Segment' columns due to more than 1490+ "unknown" values keeping the data will lead to inaccurate machine learning model

```
In [26]: df.drop(['Risk_Segment_During_Rx','Tscore_Bucket_During_Rx','Change_T_Score','Change_Risk_Segment'], axis = 1, inplace = True)
```

If there were NA values, to clean the data we would have replaced NA values with the mode or mean.

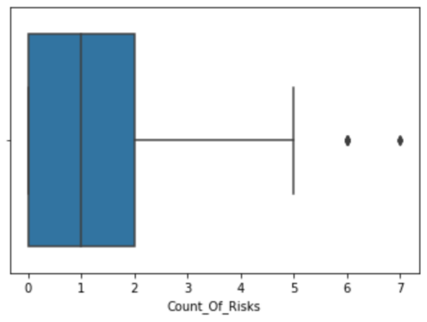
```
In [7]: df.mean()

<ipython-input-7-c61f0c8f89b5>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df.mean()

Out[7]: Dexa_Freq_During_Rx    3.016063
        Count_Of_Risks         1.239486
        dtype: float64
```

```
In [10]: df.mode()

Out[10]:
```

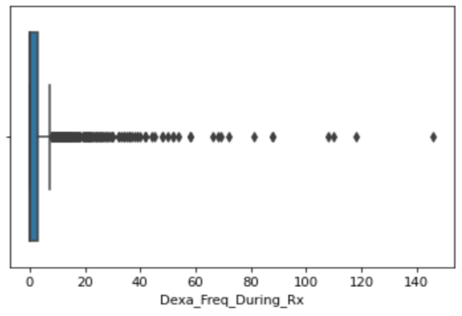
Health_Frailty	Risk_Excessive_Thinness	Risk_Hysterectomy_Oophorectomy	Risk_Estrogen_Deficiency	Risk_Immobilization	Risk_Recurring_Falls	Count_Of_Risks
N	N	N	N	N	N	1.0
NaN	NaN	NaN	NaN	NaN	NaN	NaN

Outliers – Outliers were present in both Count_Of_Risks and Dexa_Freq_During_Rx .

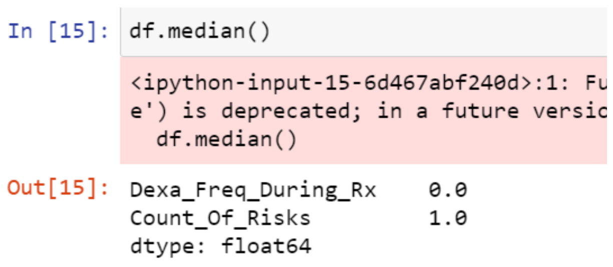
```
In [10]: sns.boxplot(x=df["Count_Of_Risks"])

Out[10]: <AxesSubplot:xlabel='Count_Of_Risks'>
```



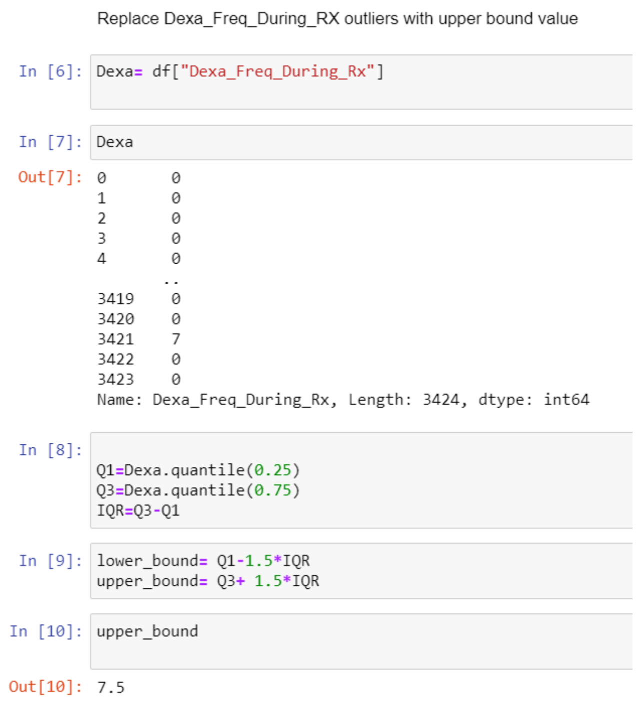
```
In [11]: sns.boxplot(x=df['Dexa_Freq_During_Rx'])

Out[11]: <AxesSubplot:xlabel='Dexa_Freq_During_Rx'>
```



Two methods were considered to deal with outliers firstly replace the outlier values to the median value or replace outlier values with upper bound value. We decided to replace outliers with upper bound values.

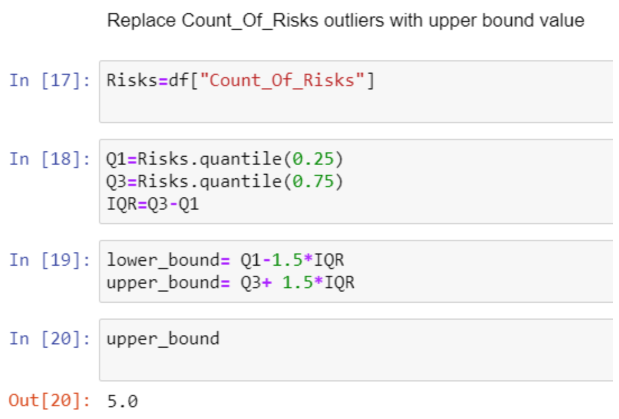


```
In [15]: df.median()

<ipython-input-15-6d467abf240d>:1: Fu
e') is deprecated; in a future versic
  df.median()

Out[15]: Dexa_Freq_During_Rx     0.0
         Count_Of_Risks         1.0
         dtype: float64
```



Replace Dexa_Freq_During_RX outliers with upper bound value

```
In [6]: Dexa= df["Dexa_Freq_During_Rx"]

In [7]: Dexa

Out[7]: 0       0
        1       0
        2       0
        3       0
        4       0
               ..
        3419    0
        3420    0
        3421    7
        3422    0
        3423    0
        Name: Dexa_Freq_During_Rx, Length: 3424, dtype: int64

In [8]:
        Q1=Dexa.quantile(0.25)
        Q3=Dexa.quantile(0.75)
        IQR=Q3-Q1

In [9]: lower_bound= Q1-1.5*IQR
        upper_bound= Q3+ 1.5*IQR

In [10]: upper_bound

Out[10]: 7.5
```

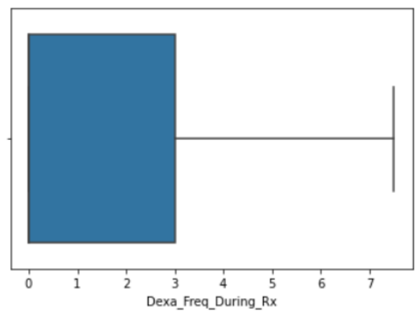


Replace Count_Of_Risks outliers with upper bound value

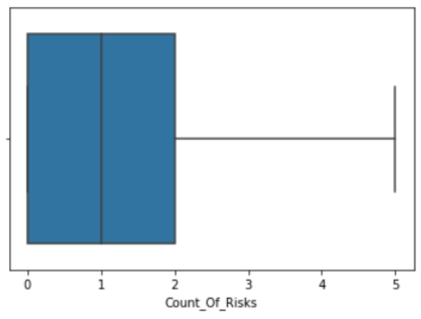
```
In [17]: Risks=df["Count_Of_Risks"]

In [18]: Q1=Risks.quantile(0.25)
         Q3=Risks.quantile(0.75)
         IQR=Q3-Q1

In [19]: lower_bound= Q1-1.5*IQR
         upper_bound= Q3+ 1.5*IQR

In [20]: upper_bound

Out[20]: 5.0
```

Boxplot after replacing outlier values.

```
In [30]: sns.boxplot(x=df['Dexa_Freq_During_Rx'])

Out[30]: <AxesSubplot:xlabel='Dexa_Freq_During_Rx'>
```

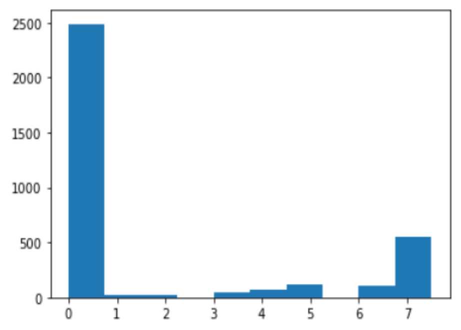


```
]: sns.boxplot(x=df["Count_Of_Risks"])

]: <AxesSubplot:xlabel='Count_Of_Risks'>
```

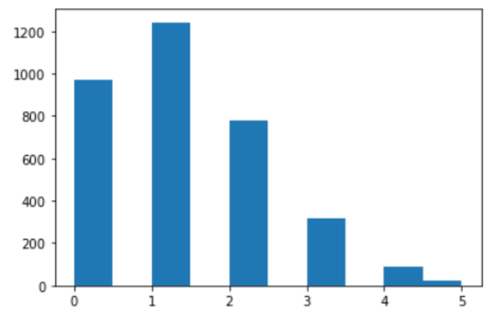


Normal distribution – the data don't follow a normal distribution both skewed to the right therefore had to be normalised.

```
In [46]: plt.hist(df["Dexa_Freq_During_Rx"])

Out[46]: (array([2488.,   24.,   24.,    0.,   46.,   68.,  114.
           553.]),
          array([0.  , 0.75, 1.5 , 2.25, 3.  , 3.75, 4.5 , 5.25,
          <BarContainer object of 10 artists>)
```



```
plt.hist(df["Count_Of_Risks"])

(array([ 970.,    0., 1242.,    0.,  781.,    0.,
         23.]),
 array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4.
 <BarContainer object of 10 artists>)
```



```
In [51]: from sklearn.preprocessing import MinMaxScaler
         normalse=MinMaxScaler()

In [52]: df[["Count_Of_Risks", "Dexa_Freq_During_Rx"]] = normalse.fit_transform(df[["Count_Of_Risks", "Dexa_Freq_During_Rx"]])
```

Data transformation - label encoding – The dataset contained categorical data which had to be converted to numerical data for the machine learning model.

```
In [74]: from sklearn.preprocessing import LabelEncoder

In [75]: final = df.apply(LabelEncoder().fit_transform)

In [76]: final

Out[76]:
```

	Persistency_Flag	Gender	Race	Ethnicity	Region	Age_Bucket	Ntm_Speciality
0	1	1	2	1	4	3	5
1	0	1	1	1	4	0	5
2	0	0	3	0	0	1	5
3	0	0	2	1	0	3	5
4	0	0	2	1	0	3	5
...	...	...	...	...	...	...	...
3419	1	0	2	1	3	3	5
3420	1	0	2	1	3	3	34
3421	1	0	2	1	3	3	3
3422	0	0	2	1	3	0	34
3423	0	0	2	1	3	1	34

3424 rows × 64 columns