Week 9– Healthcare Project

Group Name: Cool Data Scientists Team

Team Members Details:

Name	Email	Country	College / Company	Specialization
Yousef Elbayoumi	yousefxelbayomi@gmail.com	Palestine	Bahçeşehir University	Data Science
Mukhammadjon Kholmirzev	kmukhammadjon@gmail.com	Uzbekistan	Ulsan National Institute of Science and Technology	Data Science
Jamila hamdi	jamila.hamdi90@gmail.com	Tunisia	Faculty of science and managment	Data Science
H. Melis Tekin Akcin	meliss85@gmail.com	UK	Hacettepe University	Data Science

Problem Description

One of the challenges for Pharmaceutical companies is to understand the persistency of drug as per the physician prescription. This issue results in a bad impact on the pharmacies for all the categories; patients, physicians, and administration. However, the team of data scientist is capable of discovering the analyzing the dataset and detecting the factors that are impacting the primary factor which is the "persistency". By building a classification machine learning model, we will be able to classify the dataset and find the variables that affect the target variables "Persistency Flag".

Data cleansing

In the beginning, we saw many values as “Unknown” and to make the model perform better we had to remove them, the unknown values were in 4 columns which are:
'Risk_Segment_During_Rx','Tscore_Bucket_During_Rx', 'Change_T_Score' and 'Change_Risk_Segment'

```
[ ] df[df['Risk_Segment_During_Rx'] == 'Unknown']['Tscore_Bucket_During_Rx'].value_counts()
```

```
Unknown    1497
Name: Tscore_Bucket_During_Rx, dtype: int64
```

```
[ ] df[df['Change_T_Score'] == 'Unknown']['Tscore_Bucket_During_Rx'].value_counts()
```

```
Unknown    1497
Name: Tscore_Bucket_During_Rx, dtype: int64
```

```
[ ] df[df['Tscore_Bucket_During_Rx'] == 'Unknown']['Change_T_Score'].value_counts()
```

```
Unknown    1497
Name: Change_T_Score, dtype: int64
```

```
[ ] df[df['Change_Risk_Segment'] == 'Unknown']['Change_T_Score'].value_counts()
```

```
Unknown      1497
No change     601
Worsened       83
Improved       48
Name: Change_T_Score, dtype: int64
```

After checking the unknown values and their count, we did the cleaning by dropping these values:

```
[ ] #dropping columns that contains many 'Unknown' values
    df.drop(['Risk_Segment_During_Rx','Tscore_Bucket_During_Rx','Change_T_Score','Change_Risk_Segment'], axis = 1, inplace = True)
    df.head()
```

	Ptid	Persistency_Flag	Gender	Race	Ethnicity	Region	Age_Bucket	Ntm_Speciality	Ntm_Specialist_Flag	Ntm_Speciality_Bucket	Gluco_Record_Prior_Ntm	Gluco_Record_
0	P1	Persistent	Male	Caucasian	Not Hispanic	West	>75	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	N	
1	P2	Non-Persistent	Male	Asian	Not Hispanic	West	55-65	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	N	
2	P3	Non-Persistent	Female	Other/Unknown	Hispanic	Midwest	65-75	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	N	
3	P4	Non-Persistent	Female	Caucasian	Not Hispanic	Midwest	>75	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	N	
4	P5	Non-Persistent	Female	Caucasian	Not Hispanic	Midwest	>75	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	Y	

We checked also for the “Null” values, but we didn’t find any nulls in the data:

Checking Null Valeus

```
[ ] print(df.isnull().sum().sum())
```

```
0
```

As we were required to clean the data with more than one technique, we decided to clean the missing values with the mean/median/mode techniques, but as there was no null values, we just shown how these techniques work nothing more:

Cleaning data using Mean techninque

```
[ ] print("\n----------- Calculate Mean -----------\n")
    print(df.mean())
    df_mean = df
    df_mean['Count_Of_Risks'].fillna(value=df_mean['Count_Of_Risks'].mean(), inplace=True)
    df_mean['Dexa_Freq_During_Rx'].fillna(value=df_mean['Dexa_Freq_During_Rx'].mean(), inplace=True)
```

```
----------- Calculate Mean -----------

Dexa_Freq_During_Rx    3.016063
Count_Of_Risks         1.239486
dtype: float64
```

Cleaning data using Median techninque

```
[ ] print("\n----------- Calculate Median -----------\n")
    print(df.median())
    df_median = df
    df_median['Count_Of_Risks'].fillna(value=df_median['Count_Of_Risks'].mean(), inplace=True)
    df_median['Dexa_Freq_During_Rx'].fillna(value=df_median['Dexa_Freq_During_Rx'].mean(), inplace=True)
```

```
----------- Calculate Median -----------

Dexa_Freq_During_Rx    0.0
Count_Of_Risks         1.0
dtype: float64
```

Cleaning data using mode techninque

```
[ ]
    print("\n----------- Calculate Mode -----------\n")
    print(df.mode())
    df_mode = df
    df_mode['Count_Of_Risks'].fillna(value=df_mode['Count_Of_Risks'].mean(), inplace=True)
    df_mode['Dexa_Freq_During_Rx'].fillna(value=df_mode['Dexa_Freq_During_Rx'].mean(), inplace=True)
```

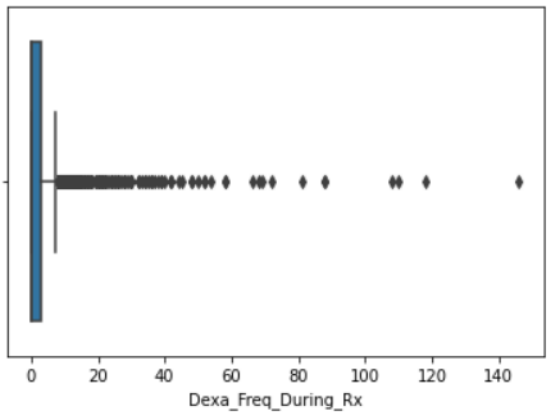
```
----------- Calculate Mode -----------

       Ptid Persistency_Flag  ... Risk_Recurring_Falls Count_Of_Risks
0        P1   Non-Persistent  ...                   N            1.0
1       P10              NaN  ...                 NaN            NaN
2      P100              NaN  ...                 NaN            NaN
3     P1000              NaN  ...                 NaN            NaN
4     P1001              NaN  ...                 NaN            NaN
...     ...              ...  ...                 ...            ...
3419   P995              NaN  ...                 NaN            NaN
3420   P996              NaN  ...                 NaN            NaN
3421   P997              NaN  ...                 NaN            NaN
3422   P998              NaN  ...                 NaN            NaN
3423   P999              NaN  ...                 NaN            NaN

[3424 rows x 65 columns]
```

We also wanted to check the outliers. However, we only have two numerical columns in the data, therefore we checked for them only, the columns are: "Dexa_Freq_During_Rx" and "Count_Of_Risks", we used the box plot and the distribution plot to check for outliers:

Checking Outliers for 1st column

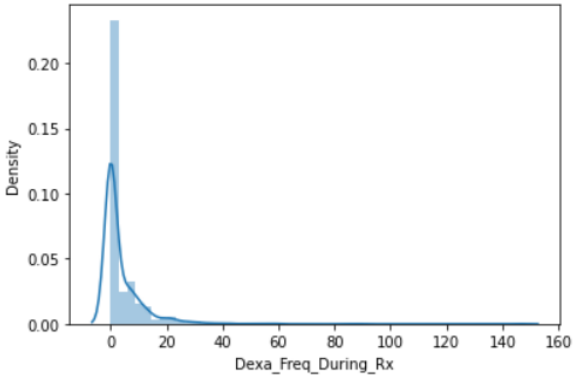
```
[ ] import seaborn as sns
    # Box plot
    sns.boxplot(df.Dexa_Freq_During_Rx)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7ff376890450>
```




```
[ ] # Distribution plot
    sns.distplot(df.Dexa_Freq_During_Rx)
```

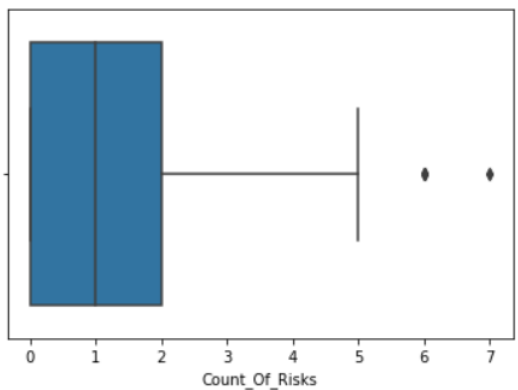
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7ff376c1e750>
```



Checking Outliers for 2nd column

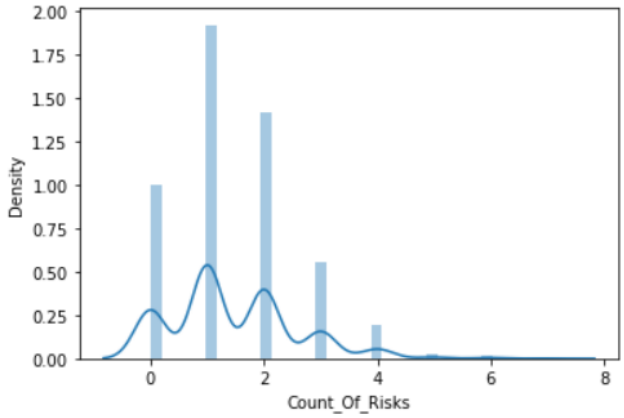
```
[ ] # Box plot
    sns.boxplot(df.Count_Of_Risks)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7ff37684b350>
```



```
[ ]  # Distribution plot
     sns.distplot(df.Count_Of_Risks)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7ff37b630210>
```



Final part was to export the data:

Exporting The Cleaned Data

```
[ ]  df.to_csv("Healthcare_cleaned_dataset.csv", index = False)
```

Data transformation

In this step, we are going to get ready our features to the machine learning model in order to predict the data. To do this, we **transform categorical data into numbers**. So, we will convert features regarding to the numbers of categories (two categories or multiple categories) . Then, we will convert Y/N values to 1/0.

- **Converting categorical variables that contains two categories using Label Encoding**

The categorical variables with two categories are : 'Persistency_Flag', 'Gender','Ntm_Specialist_Flag','Risk_Segment_Prior_Ntm','Adherent_Flag', 'Tscore_Bucket_Prior_Ntm','Gluco_Record_Prior_Ntm','Gluco_Record_During_Rx','Dexa_During_Rx','Frag_Frac_During_Rx','Adherent_Flag','Idn_Indicator','Injectable_Experience_During_Rx','Frag_Frac_Prior_Ntm'

- **Coverting categorical variables with multiple categories using Label Encoding**

The categorical variables with multiple categories are : 'Age_Bucket', 'Ntm_Speciality', 'Ntm_Speciality_Bucket','Race','Ethnicity','Region'

Now, as seen in the table below, the rest of non encoding columns categorical are Y and N values.

So, we are going to replace all o them by 1 and 0 respectively, without encode each column alone.

:h_Frailty	Risk_Excessive_Thinness	Risk_Hysterectomy_Oophorectomy	Risk_Estrogen_Deficiency	Risk_Immobilization	Risk_Recurring_Falls	Count_Of_Risks
N	N	N	N	N	N	0
N	N	N	N	N	N	0
N	N	N	N	N	N	2
N	N	N	N	N	N	1
N	N	N	N	N	N	1
...	...	...	...	...	...	...
N	N	N	N	N	N	1
N	N	N	N	N	N	0
N	N	N	N	N	N	1
N	N	N	N	N	N	0
N	N	N	N	N	N	1

- **Checking for data types after data tranformation step**

```
[ ]  df.dtypes
```

```
Ptid                                object
Persistency_Flag                     int64
Gender                               int64
Race                                 int64
Ethnicity                            int64
                                     ...
Risk_Hysterectomy_Oophorectomy       int64
Risk_Estrogen_Deficiency             int64
Risk_Immobilization                  int64
Risk_Recurring_Falls                 int64
Count_Of_Risks                       int64
Length: 65, dtype: object
```

Example of transformed features

Persistency_Flag	Gender	Race	Ethnicity	Region	Age_Bucket	Ntm_Speciality	Ntm_Specialist_Flag	Ntm_Speciality_Bucket	Gluco_Record_Prior_Ntm	C
1	1	2	1	4	3	5	0	1	0	
0	1	1	1	4	0	5	0	1	0	
0	0	3	0	0	1	5	0	1	0	
0	0	2	1	0	3	5	0	1	0	
0	0	2	1	0	3	5	0	1	1	
...	...	...	...	...	...	...	...	...	...	
1	0	2	1	3	3	5	0	1	0	
1	0	2	1	3	3	34	0	1	0	
1	0	2	1	3	3	3	1	0	0	
0	0	2	1	3	0	34	0	1	0	
0	0	2	1	3	1	34	0	1	1	

:olumns

Dexa_Freq_During_Rx	Dexa_During_Rx	Frag_Frac_Prior_Ntm	Frag_Frac_During_Rx	Risk_Segment_Prior_Ntm	Tscore_Bucket_Prior_Ntm	Adherent_Flag	Idn_Indic
0	0	0	0	1	1	0	
0	0	0	0	1	1	0	
0	0	0	0	0	0	0	
0	0	0	0	0	1	0	
0	0	0	0	0	0	0	
...	...	...	...	...	...	...	
0	0	0	0	1	1	0	
0	0	0	0	0	0	1	
7	1	0	0	1	1	0	
0	0	0	1	1	1	0	
0	0	0	0	1	1	0	

Github Repo link

https://github.com/melis-ta/Healthcare