Ali-Just Team Members

Name: Alireza Samadifardheris
Email: alirezasamadii71@gmail.com
Country: Rome, Italy
College: Sapienza University of Rome, Computer Engineering
Specialization: Data Science

Name: Justin Lee
Email: justindavinlee@gmail.com
Country: Ontario, Canada
College: Wilfrid Laurier University, Data Science Concentration Big Data
Specialization: Data Science

Link to GitHub Repository:
https://github.com/justindavin/DataGlacierInternship/tree/main/Week9

Problem Description

One of the challenges for all Pharmaceutical companies is to understand the persistence of drugs as per the physician's prescription. the persistency of a drug may be defined as "the extent to which a patient acts in accordance with the prescribed interval, and dose of a dosing regimen." Medication persistence refers to the act of continuing the treatment for the prescribed duration.

Data Cleansing and Transformation

After performing extensive exploratory data analysis on this dataset, we have found some interesting information about the data and have performed some transformations as well to improve the data intake of our model:

- One feature was mis-labelled,
  "`Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx`" and was changed to:
  "`Comorb_Encntr_For_General_Exam_W_O_Complaint_Susp_Or_Reprtd_Dx`"

- Most features are of the 'object' dtype, which are categorical, as well as strings. These features will have to be converted to numeric values or dummy variables. The way we have done this is by creating a multi column label encoder by using the LabelEncoder algorithm from sklearn.preprocessing to transform all columns into integers.

4) Data mapping

```
from sklearn.preprocessing import LabelEncoder

class MultiColLabelEncoder:
  def __init__(self,columns = None):
      self.columns = columns # array of column names to encode

  def fit(self,X,y=None):
      return self

  def transform(self,X):
      #transforms columns of X specified in self.columns using LabelEncoder().
      #If no columns specified, transforms all columns in X.

      output = X.copy()
      if self.columns is not None:
          for col in self.columns:
              output[col] = LabelEncoder().fit_transform(output[col])
      else:
          for colname,col in output.iteritems():
              output[colname] = LabelEncoder().fit_transform(col)
      return output

  def fit_transform(self,X,y=None):
      return self.fit(X,y).transform(X)
```

```
[16] df = MultiColLabelEncoder(columns= ['Ptid', 'Persistency_Flag', 'Gender', 'Race', 'Ethnicity', 'Region', 'Age_Bucket', 'Ntm_Speciality', 'Ntm_Specialist_Flag', 'Ntm_Speciality_Bucket', 'Gluco_Record_Prior_Ntm',
```

```
df.head()
```

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

- While we are tasked to look for and clean missing values, it turns out there are no missing values present in the dataset.

```
print(f"\nMissing values present in features: {df.isnull().any()}")
```

```
Missing values present in features: Ptid                          False
Persistency_Flag                 False
Gender                           False
Race                             False
Ethnicity                        False
                                 ...
Risk_Hysterectomy_Oophorectomy   False
Risk_Estrogen_Deficiency         False
Risk_Immobilization              False
Risk_Recurring_Falls             False
Count_Of_Risks                   False
Length: 69, dtype: bool
```

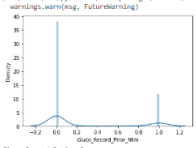
```
[11] print(f"\nMissing values count: ", df.isnull().sum())
```

```
Missing values count:  Ptid                          0
Persistency_Flag                 0
Gender                           0
Race                             0
Ethnicity                        0
                                ..
Risk_Hysterectomy_Oophorectomy   0
Risk_Estrogen_Deficiency         0
Risk_Immobilization              0
Risk_Recurring_Falls             0
Count_Of_Risks                   0
Length: 69, dtype: int64
```

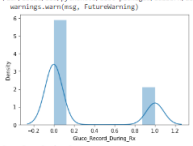
- The Ptid feature is a unique row id and will be dropped since using a unique row id as a feature in our models will result in overfitting the data.

- We have discovered that there is skewness in the data. To solve the issue of skewness, we will remove skewness from features with low correlation, as removing skew from data with higher correlation will have a higher impact and could be problematic. To remove the skewness of features with low correlation, we have first calculated the skewness of each feature. Then, we compared the values to a certain threshold of 0.01. If the correlation of the feature is below the threshold, then we remove the skewness from the features.

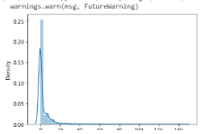
Gluco_Record_Prior_Ntm
1.2493139611257034
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)



Gluco_Record_During_Rx
1.6748863679431598
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)



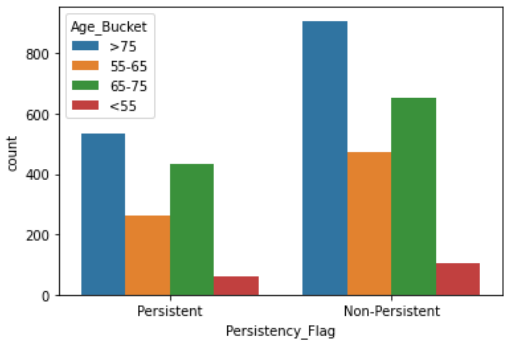
Dexa_Freq_During_Rx
6.807147951718919
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)



We have also discovered that the dataset is unbalanced. To deal with the problem of the unbalanced data, we will perform sampling techniques such as: RandomUnderSampler and SMOTE (Synthetic Minority Oversampling Technique) which will be performed in the model building portion of the project.

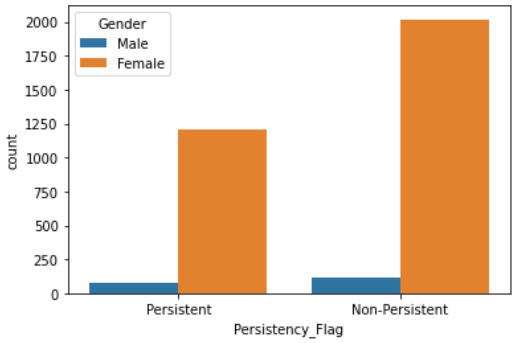
```
sns.countplot(x='Persistency_Flag', hue='Age_Bucket',  data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0ee09ced00>
```




```
sns.countplot(x='Persistency_Flag', hue='Gender', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0ee09b7e20>
```



Data Intake Report

Name: Healthcare Industry
Report date: December 2nd, 2022
Internship Batch: LISUM14
Version: 0.3
Data intake by: Alireza Samadifardheris and Justin Lee
Data intake reviewer: Alireza Samadifardheris and Justin Lee
Data storage location:
https://docs.google.com/spreadsheets/d/1P_oMc6gOBlhw6dY5PxaqxV2swdHMUooK/edit#gid=2047360270

Tabular data details:

Total number of observations	3424
Total number of files	
Total number of features	69
Base format of the file	.xlsx
Size of the data	900 KB

Note: Convert this doc in pdf and provide the link of pdf file in your dashboard.
Please do not forget to remove this section while converting the file into pdf.