Data Intake Report - Data Science Healthcare - Persistency of a Drug – Classification – Week 9

Group Name: <Igor Azevedo de Queiroz>
Email: <igor_queiroz17@yahoo.com.br>
Country: <Ireland>
College: <Dublin Business School>
Specialization: <Data Science - Classification>
Report date: <20/09/2021>
Internship Batch:<LISUM02>
Version:<1.0>
Data storage location: https://github.com/IgorQueiroz32/Data-Science-Healthcare---Persistency-of-a-Drug-Classification/tree/main/week%209

Problem Description: <Data Science Healthcare - Persistency of a Drug - Classification>

Tabular data details:

Total number of observations	< 3424rows >
Total number of files	<1>
Total number of features	< 69 columns >
Base format of the file	<.ipynb, .csv, .txt, .png>
Size of the data	<891 in KB>

Healthcare - Persistency of a Drug - Classification

1. Business Description/ Problem.

One of the challenges for all Pharmaceutical companies is to understand the persistency of drug as per the physician prescription. To solve this problem ABC pharma company approached an analytics company to automate this process of identification.

With an objective to gather insights on the factors that are impacting the persistency, it is necessary to build a classification for the given dataset, using the variable 'Persistency_Flag' as target variable and other attributes as prediction variables.

2. Business Understanding.

ABC it is a private pharma company. Due to the problem to the persistency of drug as per the physician prescription, a data science project is applied to predict the classification of 'Persistency_Flag' variable. In other words, based on the previously patients characteristics it is possible predict if futures patients will use the drugs during the role treatment or if they won't.

The object of this project is providing answer of the main questions made by the company's CEO, which are:

- What is the 'Persistency_Flag' classification for future patients?

The answer for those questions is presented in two different methods:

- A webapp with all necessary prediction attributes in order to predict the classification of the 'Persistency_Flag' for future patients.
- A dashboard with several hypotheses and insights to help the company CEO with future decisions.

The tools used for this project are: Python 3.8, Pycharm, Jupyter Notebook, Streamlit and Heroku.

3. Data Understanding.

There is 1 dataset provided:

https://www.kaggle.com/harbhajansingh21/persistent-vs-nonpersistent

healthcare_dataset.csv – this file includes characteristics of several patients.

Variables Description:

Here I'm describing the columns in detail:

Patient Details:

- **Patient ID:** Unique ID of each patient;
- **Persistency_Flag:** Flag indicating if a patient was persistent or not;
- **Age:** Age of the patient during their therapy;
- **Race:** Race of the patient from the patient table;
- **Region:** Region of the patient from the patient table;
- **Ethnicity:** Ethnicity of the patient from the patient table;
- **Gender:** Gender of the patient from the patient table;
- **IDN Indicator:** Flag indicating patients mapped to IDN;

Provider Attributes:

- **NTM - Physician Specialty:** Specialty of the HCP that prescribed the NTM Rx;

Clinical Factors:

- **NTM - T-Score:** T Score of the patient at the time of the NTM Rx (within 2 years prior from rxdate);
- **Change in T Score:** Change in Tscore before starting with any therapy and after receiving therapy (Worsened, Remained Same, Improved, Unknown);
- **NTM - Risk Segment:** Risk Segment of the patient at the time of the NTM Rx (within 2 years days prior from rxdate);

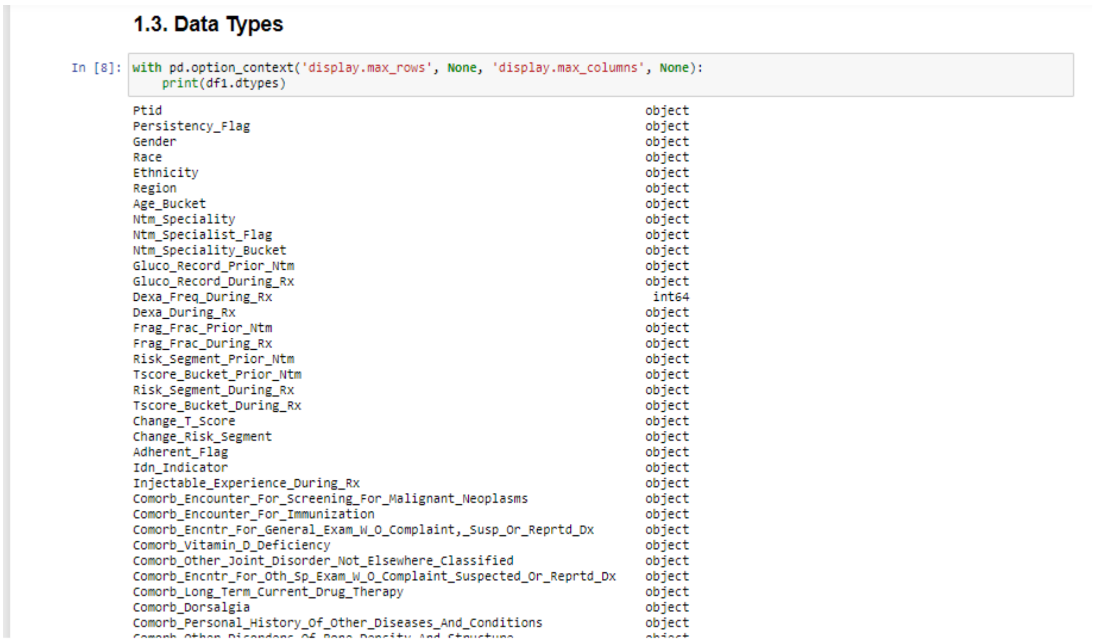
- **Change in Risk Segment:** Change in Risk Segment before starting with any therapy and after receiving therapy (Worsened, Remained Same, Improved, Unknown);
- **NTM - Multiple Risk Factors:** Flag indicating if patient falls under multiple risk category (having more than 1 risk) at the time of the NTM Rx (within 365 days prior from rxdate);
- **NTM - Dexa Scan Frequency:** Number of DEXA scans taken prior to the first NTM Rx date (within 365 days prior from rxdate);
- **NTM - Dexa Scan Recency:** Flag indicating the presence of Dexa Scan before the NTM Rx (within 2 years prior from rxdate or between their first Rx and Switched Rx; whichever is smaller and applicable);
- **Dexa During Therapy:** Flag indicating if the patient had a Dexa Scan during their first continuous therapy;
- **NTM - Fragility Fracture Recency:** Flag indicating if the patient had a recent fragility fracture (within 365 days prior from rxdate);
- **Fragility Fracture During Therapy**: Flag indicating if the patient had fragility fracture during their first continuous therapy;
- **NTM - Glucocorticoid Recency:** Flag indicating usage of Glucocorticoids (>=7.5mg strength) in the one year look-back from the first NTM Rx;
- **Glucocorticoid During Therapy:** Flag indicating if the patient had a Glucocorticoid usage during the first continuous therapy;

Disease/Treatment Factors:

- **NTM - Injectable Experience:** Flag indicating any injectable drug usage in the recent 12 months before the NTM OP Rx;
- **NTM - Risk Factors:** Risk Factors that the patient is falling into. For chronic Risk Factors complete lookback to be applied and for non-chronic Risk Factors, one year lookback from the date of first OP Rx;
- **NTM - Comorbidity:** Comorbidities are divided into two main categories - Acute and chronic, based on the ICD codes. For chronic disease we are taking complete look back from the first Rx date of NTM therapy and for acute diseases, time period before the NTM OP Rx with one year lookback has been applied;
- **NTM - Concomitancy:** Concomitant drugs recorded prior to starting with a therapy (within 365 days prior from first rxdate)
  Adherence: Adherence for the therapies.

4. Data Type.

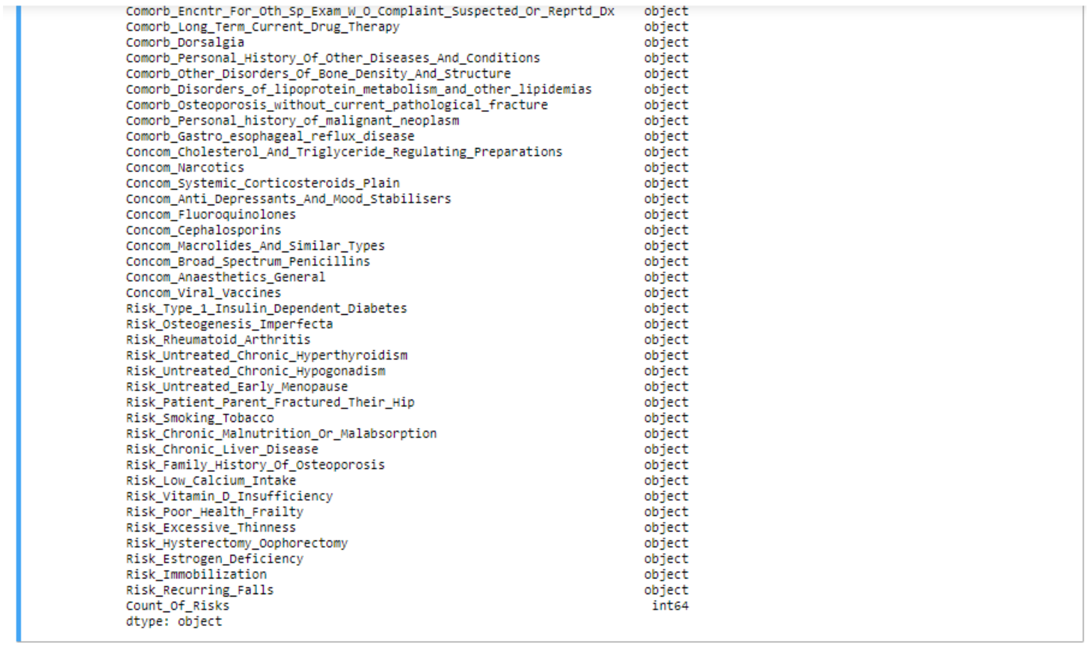
The majority of the attributes of this dataset is from type object, initially just 2 attributes is type int64.



1.3. Data Types

```
In [8]: with pd.option_context('display.max_rows', None, 'display.max_columns', None):
            print(df1.dtypes)
```

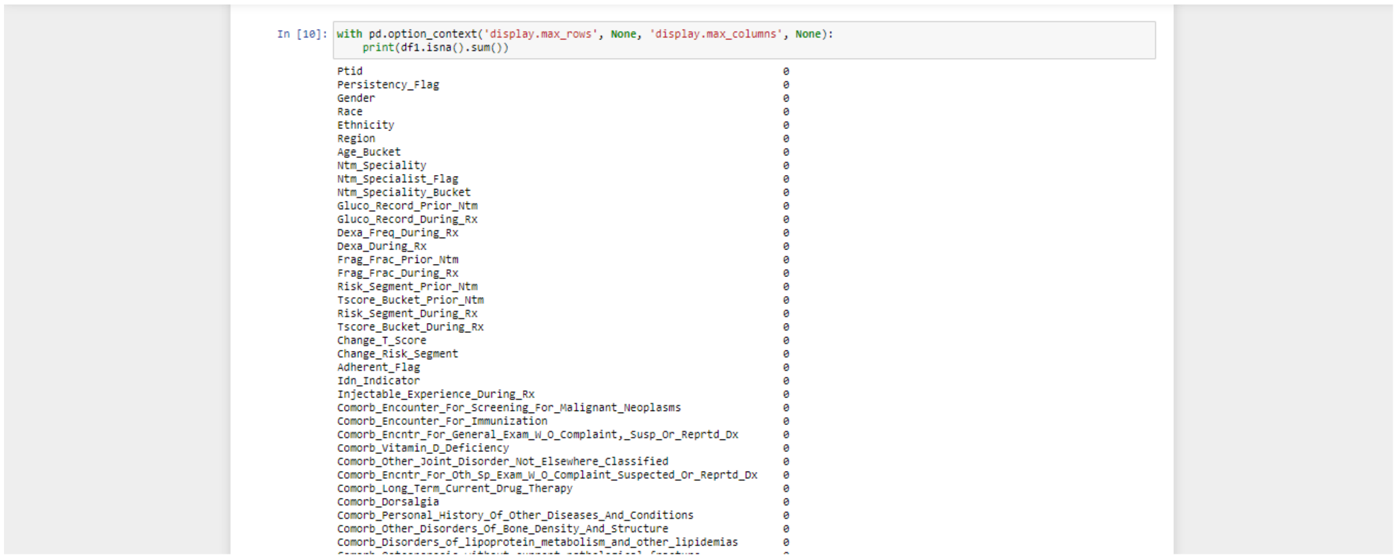
```
Ptid                                                                  object
Persistency_Flag                                                      object
Gender                                                                object
Race                                                                  object
Ethnicity                                                             object
Region                                                                object
Age_Bucket                                                            object
Ntm_Speciality                                                        object
Ntm_Specialist_Flag                                                   object
Ntm_Speciality_Bucket                                                 object
Gluco_Record_Prior_Ntm                                                object
Gluco_Record_During_Rx                                                object
Dexa_Freq_During_Rx                                                    int64
Dexa_During_Rx                                                        object
Frag_Frac_Prior_Ntm                                                   object
Frag_Frac_During_Rx                                                   object
Risk_Segment_Prior_Ntm                                                object
Tscore_Bucket_Prior_Ntm                                               object
Risk_Segment_During_Rx                                                object
Tscore_Bucket_During_Rx                                               object
Change_T_Score                                                        object
Change_Risk_Segment                                                   object
Adherent_Flag                                                         object
Idn_Indicator                                                         object
Injectable_Experience_During_Rx                                       object
Comorb_Encounter_For_Screening_For_Malignant_Neoplasms                object
Comorb_Encounter_For_Immunization                                     object
Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx       object
Comorb_Vitamin_D_Deficiency                                           object
Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified                  object
Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx    object
Comorb_Long_Term_Current_Drug_Therapy                                 object
Comorb_Dorsalgia                                                      object
Comorb_Personal_History_Of_Other_Diseases_And_Conditions              object
```



```
Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx    object
Comorb_Long_Term_Current_Drug_Therapy                                 object
Comorb_Dorsalgia                                                      object
Comorb_Personal_History_Of_Other_Diseases_And_Conditions              object
Comorb_Other_Disorders_Of_Bone_Density_And_Structure                  object
Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias       object
Comorb_Osteoporosis_without_current_pathological_fracture             object
Comorb_Personal_history_of_malignant_neoplasm                         object
Comorb_Gastro_esophageal_reflux_disease                               object
Concom_Cholesterol_And_Triglyceride_Regulating_Preparations           object
Concom_Narcotics                                                      object
Concom_Systemic_Corticosteroids_Plain                                 object
Concom_Anti_Depressants_And_Mood_Stabilisers                          object
Concom_Fluoroquinolones                                               object
Concom_Cephalosporins                                                 object
Concom_Macrolides_And_Similar_Types                                   object
Concom_Broad_Spectrum_Penicillins                                     object
Concom_Anaesthetics_General                                           object
Concom_Viral_Vaccines                                                 object
Risk_Type_1_Insulin_Dependent_Diabetes                                object
Risk_Osteogenesis_Imperfecta                                          object
Risk_Rheumatoid_Arthritis                                             object
Risk_Untreated_Chronic_Hyperthyroidism                                object
Risk_Untreated_Chronic_Hypogonadism                                   object
Risk_Untreated_Early_Menopause                                        object
Risk_Patient_Parent_Fractured_Their_Hip                               object
Risk_Smoking_Tobacco                                                  object
Risk_Chronic_Malnutrition_Or_Malabsorption                            object
Risk_Chronic_Liver_Disease                                            object
Risk_Family_History_Of_Osteoporosis                                   object
Risk_Low_Calcium_Intake                                               object
Risk_Vitamin_D_Insufficiency                                          object
Risk_Poor_Health_Frailty                                              object
Risk_Excessive_Thinness                                               object
Risk_Hysterectomy_Oophorectomy                                        object
Risk_Estrogen_Deficiency                                              object
Risk_Immobilization                                                   object
Risk_Recurring_Falls                                                  object
Count_Of_Risks                                                         int64
dtype: object
```

5. Dataset Problems.

The dataset have not presented problems of missing values, as it is possible to see on the picture bellow.

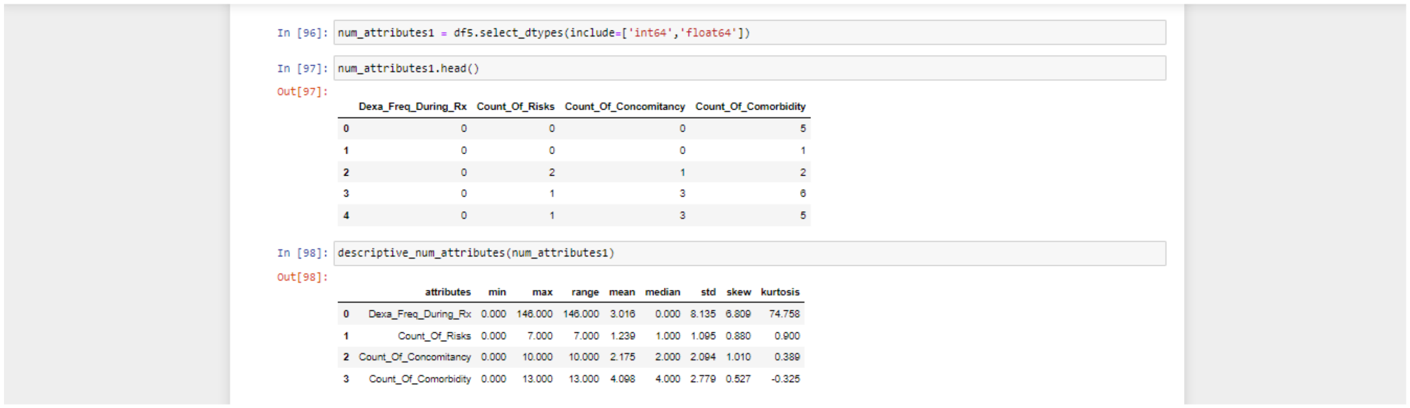


```
In [10]: with pd.option_context('display.max_rows', None, 'display.max_columns', None):
             print(df1.isna().sum())
```

```
Ptid                                                                  0
Persistency_Flag                                                      0
Gender                                                                0
Race                                                                  0
Ethnicity                                                             0
Region                                                                0
Age_Bucket                                                            0
Ntm_Speciality                                                        0
Ntm_Specialist_Flag                                                   0
Ntm_Speciality_Bucket                                                 0
Gluco_Record_Prior_Ntm                                                0
Gluco_Record_During_Rx                                                0
Dexa_Freq_During_Rx                                                   0
Dexa_During_Rx                                                        0
Frag_Frac_Prior_Ntm                                                   0
Frag_Frac_During_Rx                                                   0
Risk_Segment_Prior_Ntm                                                0
Tscore_Bucket_Prior_Ntm                                               0
Risk_Segment_During_Rx                                                0
Tscore_Bucket_During_Rx                                               0
Change_T_Score                                                        0
Change_Risk_Segment                                                   0
Adherent_Flag                                                         0
Idn_Indicator                                                         0
Injectable_Experience_During_Rx                                       0
Comorb_Encounter_For_Screening_For_Malignant_Neoplasms                0
Comorb_Encounter_For_Immunization                                     0
Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx       0
Comorb_Vitamin_D_Deficiency                                           0
Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified                  0
Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx    0
Comorb_Long_Term_Current_Drug_Therapy                                 0
Comorb_Dorsalgia                                                      0
Comorb_Personal_History_Of_Other_Diseases_And_Conditions              0
Comorb_Other_Disorders_Of_Bone_Density_And_Structure                  0
Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias       0
```

However, it presented few problems, such as:

- Higher skew and kurtosis for the variable 'Dexa_Freq_During_Rx';



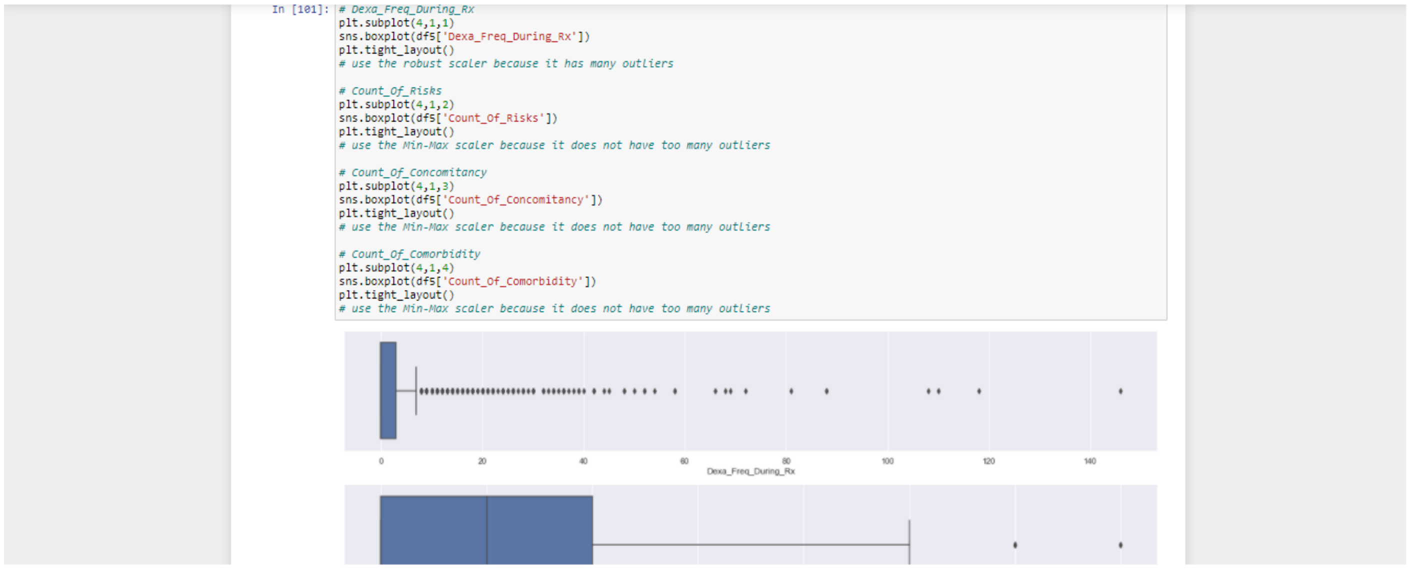
```
In [96]: num_attributes1 = df5.select_dtypes(include=['int64','float64'])

In [97]: num_attributes1.head()
Out[97]:
```

	Dexa_Freq_During_Rx	Count_Of_Risks	Count_Of_Concomitancy	Count_Of_Comorbidity
0	0	0	0	5
1	0	0	0	1
2	0	2	1	2
3	0	1	3	6
4	0	1	3	5

```
In [98]: descriptive_num_attributes(num_attributes1)
Out[98]:
```

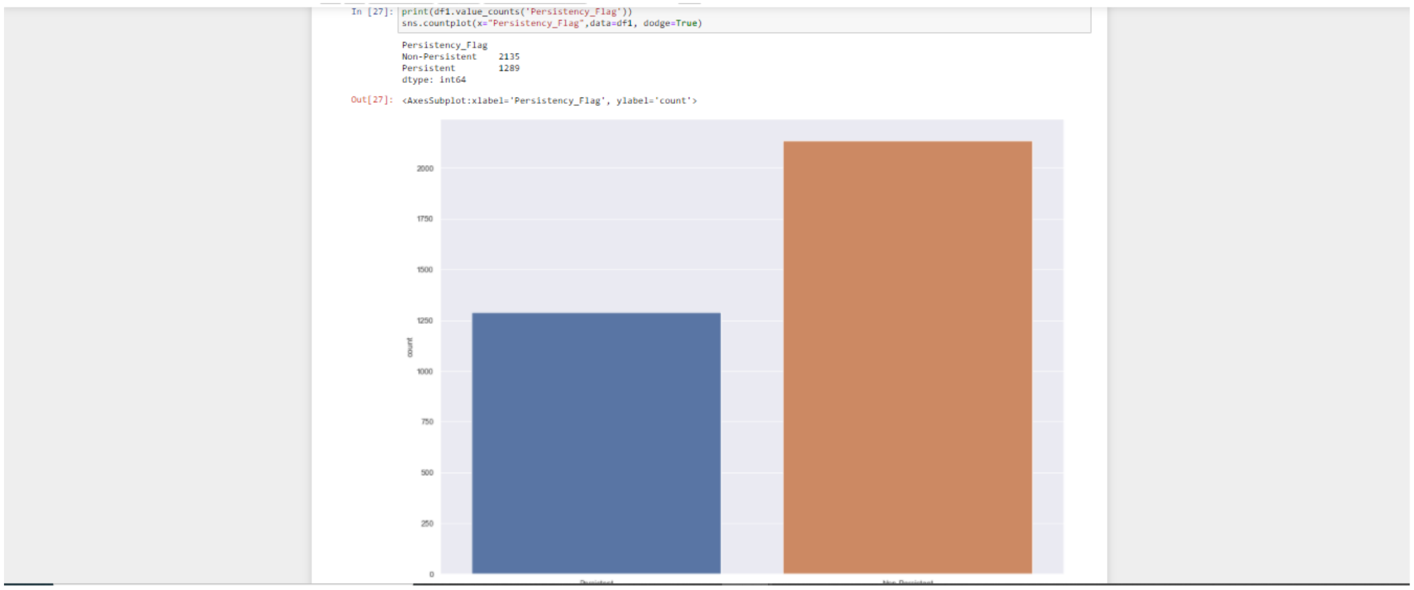
	attributes	min	max	range	mean	median	std	skew	kurtosis
0	Dexa_Freq_During_Rx	0.000	146.000	146.000	3.016	0.000	8.135	6.809	74.758
1	Count_Of_Risks	0.000	7.000	7.000	1.239	1.000	1.095	0.880	0.900
2	Count_Of_Concomitancy	0.000	10.000	10.000	2.175	2.000	2.094	1.010	0.389
3	Count_Of_Comorbidity	0.000	13.000	13.000	4.098	4.000	2.779	0.527	-0.325

- Several outliers for the variable ‘’Dexa_Freq_During_Rx’;



```
In [101]: # Dexa_Freq_During_Rx
          plt.subplot(4,1,1)
          sns.boxplot(df5['Dexa_Freq_During_Rx'])
          plt.tight_layout()
          # use the robust scaler because it has many outliers

          # Count_Of_Risks
          plt.subplot(4,1,2)
          sns.boxplot(df5['Count_Of_Risks'])
          plt.tight_layout()
          # use the Min-Max scaler because it does not have too many outliers

          # Count_Of_Concomitancy
          plt.subplot(4,1,3)
          sns.boxplot(df5['Count_Of_Concomitancy'])
          plt.tight_layout()
          # use the Min-Max scaler because it does not have too many outliers

          # Count_Of_Comorbidity
          plt.subplot(4,1,4)
          sns.boxplot(df5['Count_Of_Comorbidity'])
          plt.tight_layout()
          # use the Min-Max scaler because it does not have too many outliers
```

- Target variable unbalanced, for the target variable Persistency_Flag.



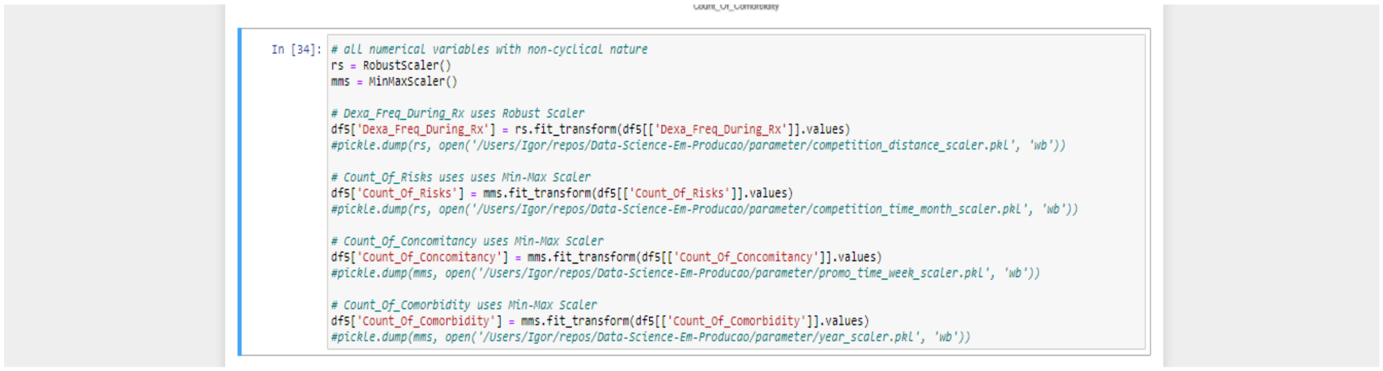
```
In [27]: print(df1.value_counts('Persistency_Flag'))
         sns.countplot(x="Persistency_Flag",data=df1, dodge=True)

         Persistency_Flag
         Non-Persistent    2135
         Persistent        1289
         dtype: int64

Out[27]: <AxesSubplot:xlabel='Persistency_Flag', ylabel='count'>
```

6. Solving the Dataset Problems.

To solve the dataset problems different approaches for each problem was taken.

For the higher skew and kurtosis for the variable ‘Dexa_Freq_During_Rx’ and for the several outliers for the variable ‘‘Dexa_Freq_During_Rx’, one step was taken:

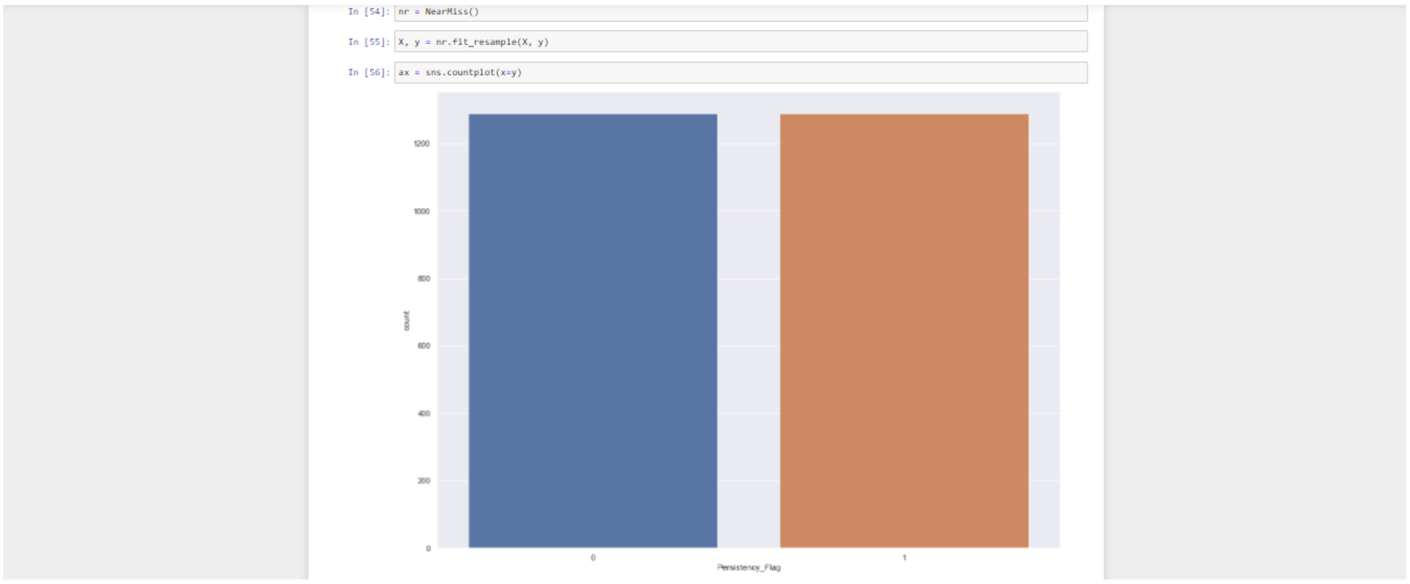
Rescaling all the numerical variables. For variables with a lot of outliers the Robust Scaler was used, for variables that do not have a lot of outliers, Min-Max Scaler was used.



```
In [34]: # all numerical variables with non-cyclical nature
         rs = RobustScaler()
         mms = MinMaxScaler()

         # Dexa_Freq_During_Rx uses Robust Scaler
         df5['Dexa_Freq_During_Rx'] = rs.fit_transform(df5[['Dexa_Freq_During_Rx']].values)
         #pickle.dump(rs, open('/Users/Igor/repos/Data-Science-Em-Producao/parameter/competition_distance_scaler.pkl', 'wb'))

         # Count_Of_Risks uses uses Min-Max Scaler
         df5['Count_Of_Risks'] = mms.fit_transform(df5[['Count_Of_Risks']].values)
         #pickle.dump(rs, open('/Users/Igor/repos/Data-Science-Em-Producao/parameter/competition_time_month_scaler.pkl', 'wb'))

         # Count_Of_Concomitancy uses Min-Max Scaler
         df5['Count_Of_Concomitancy'] = mms.fit_transform(df5[['Count_Of_Concomitancy']].values)
         #pickle.dump(mms, open('/Users/Igor/repos/Data-Science-Em-Producao/parameter/promo_time_week_scaler.pkl', 'wb'))

         # Count_Of_Comorbidity uses Min-Max Scaler
         df5['Count_Of_Comorbidity'] = mms.fit_transform(df5[['Count_Of_Comorbidity']].values)
         #pickle.dump(mms, open('/Users/Igor/repos/Data-Science-Em-Producao/parameter/year_scaler.pkl', 'wb'))
```

For the Target variable unbalanced, for the target variable Persistency_Flag, another step was taken:

Apply the function NearMiss() to reduce the size of the class with more values (Non-Persistent), and match the same class with the class with fewer values (Persistent).



```
In [54]: nr = NearMiss()

In [55]: X, y = nr.fit_resample(X, y)

In [56]: ax = sns.countplot(x=y)
```

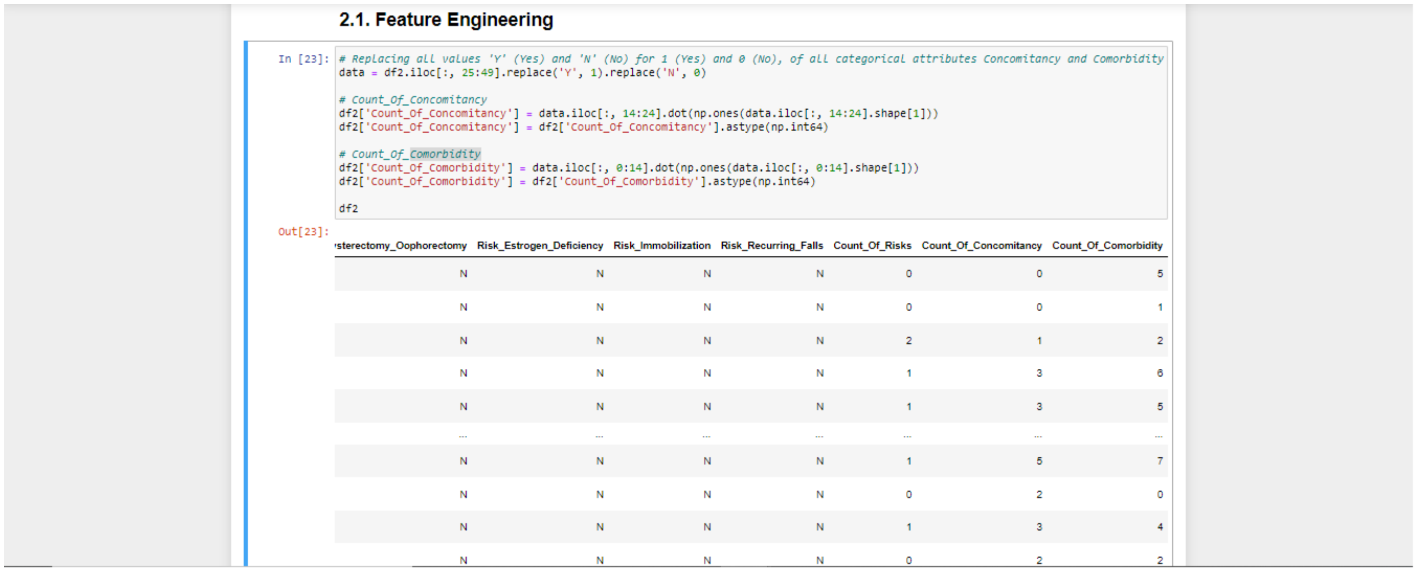
7. Feature Engineering.

Here news attributes are created to help increase the ML model and answer some hypotheses questions.

Two new features are created:

- Count_Of_Concomitancy: The total of Concomitancy that each patient presents.
- Count_Of_Comorbidity: The total of Comorbidity that each patient presents.

To create those attributes it is necessary transform all values ‘Y’ and ‘N’ to 1 and 0 respectively. Then you just sum all the values 1 for each patient.



2.1. Feature Engineering

```
In [23]: # Replacing all values 'Y' (Yes) and 'N' (No) for 1 (Yes) and 0 (No), of all categorical attributes Concomitancy and Comorbidity
         data = df2.iloc[:, 25:49].replace('Y', 1).replace('N', 0)

         # Count_Of_Concomitancy
         df2['Count_Of_Concomitancy'] = data.iloc[:, 14:24].dot(np.ones(data.iloc[:, 14:24].shape[1]))
         df2['Count_Of_Concomitancy'] = df2['Count_Of_Concomitancy'].astype(np.int64)

         # Count_Of_Comorbidity
         df2['Count_Of_Comorbidity'] = data.iloc[:, 0:14].dot(np.ones(data.iloc[:, 0:14].shape[1]))
         df2['Count_Of_Comorbidity'] = df2['Count_Of_Comorbidity'].astype(np.int64)

         df2
```

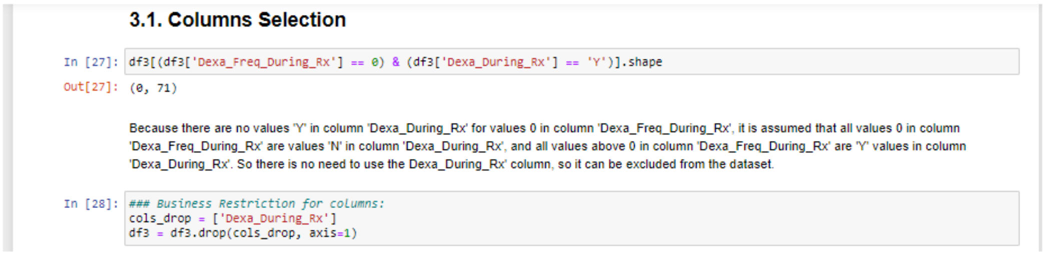
Out[23]:

sterectomy_Oophorectomy	Risk_Estrogen_Deficiency	Risk_Immobilization	Risk_Recurring_Falls	Count_Of_Risks	Count_Of_Concomitancy	Count_Of_Comorbidity
N	N	N	N	0	0	5
N	N	N	N	0	0	1
N	N	N	N	2	1	2
N	N	N	N	1	3	6
N	N	N	N	1	3	5
...	...	...	...	...	...	...
N	N	N	N	1	5	7
N	N	N	N	0	2	0
N	N	N	N	1	3	4
N	N	N	N	0	2	2

8. Columns Selections.

Here some columns are evaluated to see if they are important for the ML model or not.

The attribute 'Dexa_During_Rx' is not necessary for the model, because its information it is already included on the variable 'Dexa_Freq_During_Rx'.



3.1. Columns Selection

```
In [27]: df3[(df3['Dexa_Freq_During_Rx'] == 0) & (df3['Dexa_During_Rx'] == 'Y')].shape
Out[27]: (0, 71)
```

Because there are no values 'Y' in column 'Dexa_During_Rx' for values 0 in column 'Dexa_Freq_During_Rx', it is assumed that all values 0 in column 'Dexa_Freq_During_Rx' are values 'N' in column 'Dexa_During_Rx', and all values above 0 in column 'Dexa_Freq_During_Rx' are 'Y' values in column 'Dexa_During_Rx'. So there is no need to use the Dexa_During_Rx' column, so it can be excluded from the dataset.

```
In [28]: ### Business Restriction for columns:
         cols_drop = ['Dexa_During_Rx']
         df3 = df3.drop(cols_drop, axis=1)
```

9. Transformation.

Here every categorical attribute is transformed into numerical attribute in order to apply the ML model to make the classification.

There are several types of encoder to transform the variables, and each variable uses a unique encoder that is most suitable for this variable.

The specification of the types of encoder is listed below:

- Categorical attributes that presents binary values as 'Y' and 'N', the method **Label Encoding** will be used in order to transform 'Y' and 'N' values into 1 and 0 respectively.

- Categorical attributes that presents binary values also will be use the method **Label Encoding.**

  # Persistency_Flag (Persistent = 1, Non-Persistent = 0);
  # Gender (Male = 1, Female = 0);
  # Ntm_Specialist_Flag (Specialist = 1, Others = 0);
  # Risk_Segment_Prior_Ntm (VLR_LR = 1, HR_VHR = 0);
  # Adherent_Flag (Non-Adherent = 1, Adherent = 0);

- Categorical attributes that presents order or scale will be use the method **Ordinal Encoding.**

  # Age_Bucket;
  # Tscore_Bucket_Prior_Ntm (>-2.5 = 1, <=-2.5 = 0);

- Categorical attributes that do not presents order or scale or idea os state, each value is independent, will be use the method **Label Encoding.**

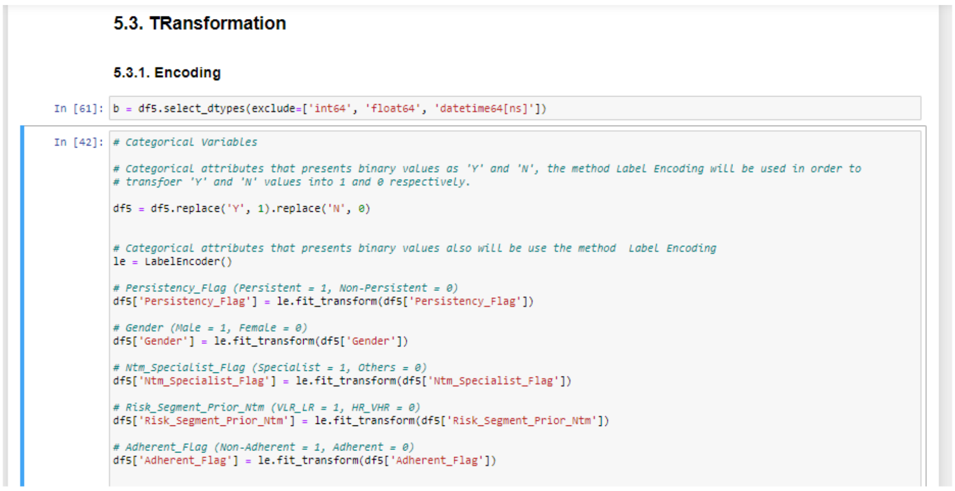
  # Race;
  # Ethnicity;
  # Region;
  # Ntm_Speciality_Bucket;
  # Risk_Segment_During_Rx;
  # Tscore_Bucket_During_Rx;

- Categorical attributes that presents an idea os state, will be use the method **One Hot Encoding.**

  # Change_T_Score;
  # Change_Risk_Segment;

- # Categorical attributes that presents a huge amount of values, will be use the method **Target Encoding.**

  # Ntm_Speciality.

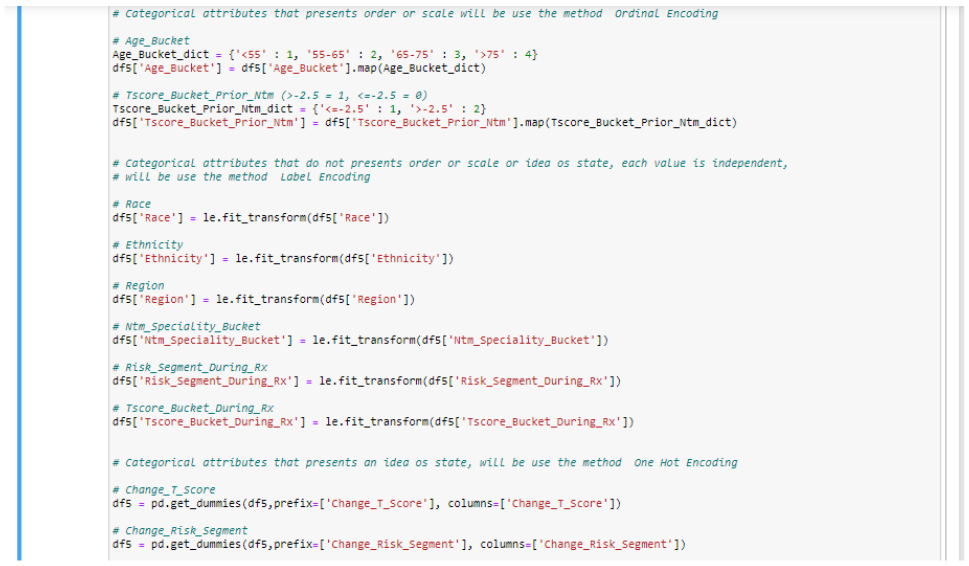


5.3. TRansformation

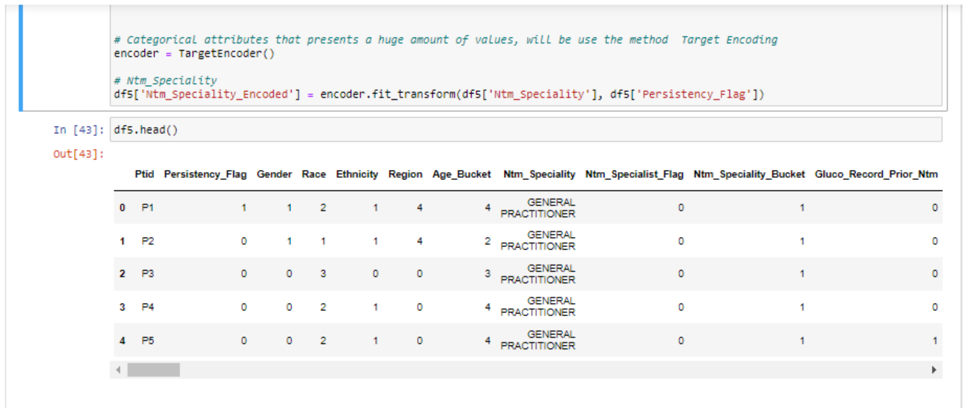
5.3.1. Encoding

```
In [61]: b = df5.select_dtypes(exclude=['int64', 'float64', 'datetime64[ns]'])
```

```
In [42]: # Categorical Variables

         # Categorical attributes that presents binary values as 'Y' and 'N', the method Label Encoding will be used in order to
         # transfoer 'Y' and 'N' values into 1 and 0 respectively.

         df5 = df5.replace('Y', 1).replace('N', 0)


         # Categorical attributes that presents binary values also will be use the method  Label Encoding
         le = LabelEncoder()

         # Persistency_Flag (Persistent = 1, Non-Persistent = 0)
         df5['Persistency_Flag'] = le.fit_transform(df5['Persistency_Flag'])

         # Gender (Male = 1, Female = 0)
         df5['Gender'] = le.fit_transform(df5['Gender'])

         # Ntm_Specialist_Flag (Specialist = 1, Others = 0)
         df5['Ntm_Specialist_Flag'] = le.fit_transform(df5['Ntm_Specialist_Flag'])

         # Risk_Segment_Prior_Ntm (VLR_LR = 1, HR_VHR = 0)
         df5['Risk_Segment_Prior_Ntm'] = le.fit_transform(df5['Risk_Segment_Prior_Ntm'])

         # Adherent_Flag (Non-Adherent = 1, Adherent = 0)
         df5['Adherent_Flag'] = le.fit_transform(df5['Adherent_Flag'])
```



```
# Categorical attributes that presents order or scale will be use the method  Ordinal Encoding

# Age_Bucket
Age_Bucket_dict = {'<55' : 1, '55-65' : 2, '65-75' : 3, '>75' : 4}
df5['Age_Bucket'] = df5['Age_Bucket'].map(Age_Bucket_dict)

# Tscore_Bucket_Prior_Ntm (>-2.5 = 1, <=-2.5 = 0)
Tscore_Bucket_Prior_Ntm_dict = {'<=-2.5' : 1, '>-2.5' : 2}
df5['Tscore_Bucket_Prior_Ntm'] = df5['Tscore_Bucket_Prior_Ntm'].map(Tscore_Bucket_Prior_Ntm_dict)


# Categorical attributes that do not presents order or scale or idea os state, each value is independent,
# will be use the method  Label Encoding

# Race
df5['Race'] = le.fit_transform(df5['Race'])

# Ethnicity
df5['Ethnicity'] = le.fit_transform(df5['Ethnicity'])

# Region
df5['Region'] = le.fit_transform(df5['Region'])

# Ntm_Speciality_Bucket
df5['Ntm_Speciality_Bucket'] = le.fit_transform(df5['Ntm_Speciality_Bucket'])

# Risk_Segment_During_Rx
df5['Risk_Segment_During_Rx'] = le.fit_transform(df5['Risk_Segment_During_Rx'])

# Tscore_Bucket_During_Rx
df5['Tscore_Bucket_During_Rx'] = le.fit_transform(df5['Tscore_Bucket_During_Rx'])


# Categorical attributes that presents an idea os state, will be use the method  One Hot Encoding

# Change_T_Score
df5 = pd.get_dummies(df5,prefix=['Change_T_Score'], columns=['Change_T_Score'])

# Change_Risk_Segment
df5 = pd.get_dummies(df5,prefix=['Change_Risk_Segment'], columns=['Change_Risk_Segment'])
```



```
# Categorical attributes that presents a huge amount of values, will be use the method  Target Encoding
encoder = TargetEncoder()

# Ntm_Speciality
df5['Ntm_Speciality_Encoded'] = encoder.fit_transform(df5['Ntm_Speciality'], df5['Persistency_Flag'])
```

```
In [43]: df5.head()
Out[43]:
```

	Ptid	Persistency_Flag	Gender	Race	Ethnicity	Region	Age_Bucket	Ntm_Speciality	Ntm_Specialist_Flag	Ntm_Speciality_Bucket	Gluco_Record_Prior_Ntm
0	P1	1	1	2	1	4	4	GENERAL PRACTITIONER	0	1	0
1	P2	0	1	1	1	4	2	GENERAL PRACTITIONER	0	1	0
2	P3	0	0	3	0	0	3	GENERAL PRACTITIONER	0	1	0
3	P4	0	0	2	1	0	4	GENERAL PRACTITIONER	0	1	0
4	P5	0	0	2	1	0	4	GENERAL PRACTITIONER	0	1	1

10. Project lifecycle along with deadline.

- Problem understanding
- Data Understanding
- Data Cleaning and Feature engineering
- Model Development
- Model Selection
- Model Evaluation

All those steps are done.