Data Science Internship

Week 13: Data Science Project

Healthcare – Persistency of the drug

Project Report

Group Name: One
Name: Hira Fahim
Email: hirashahidd26@yahoo.com
Country: United Kingdom
Company: Unemployed
Specialization: Data Science
Batch Code: LISUM19
Submission Date: 12th May 2023

Table of Contents

1. Problem Description

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

2. Business understanding

2.1. Objectives

The goal is to build a binary classification model to gather insights on the factors that are impacting the persistency of the drug.

2.2. Strategy

The analysis consists of four parts:

- Problem understanding
- Data Understanding
- Data Cleaning and Feature engineering
- Model Development
- Model Selection
- Model Evaluation
- Report the accuracy, precision, and recall of target variable.
- Report ROC-AUC
- Model Deployment
- Explain the challenges and model selection.

3. Dataset Information

The dataset consists of information about patients, doctors' speciality, clinical factors, and disease/treatment factors. All these factors have impact on persistency of the drug. The dataset consists of 69 variables including target variable and total number of 3424 observations.

4. Attribute Information

4.1. Input Variables

4.1.1. Numerical Attributes

- Dexa_Freq_During_Rx
- Count_Of_Risks

4.1.2. Categorical Attributes

- Gender
- Race
- Ethnicity
- Region
- Age_Bucket
- Ntm_Speciality
- Ntm_Specialist_Flag
- Ntm_Speciality_Bucket
- Gluco_Record_Prior_Ntm

- Gluco_Record_During_Rx
- Dexa_During_Rx
- Frag_Frac_Prior_Ntm
- Frag_Frac_During_Rx
- Risk_Segment_Prior_Ntm
- Tscore_Bucket_Prior_Ntm
- Risk_Segment_During_Rx
- Tscore_Bucket_During_Rx
- Change_T_Score
- Change_Risk_Segment
- Adherent_Flag
- Idn_Indicator
- Injectable_Experience_During_Rx
- Comorb_Encounter_For_Screening_For_Malignant_Neoplasms
- Comorb_Encounter_For_Immunization
- Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx
- Comorb_Vitamin_D_Deficiency
- Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified
- Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx
- Comorb_Long_Term_Current_Drug_Therapy
- Comorb_Dorsalgia
- Comorb_Personal_History_Of_Other_Diseases_And_Conditions
- Comorb_Other_Disorders_Of_Bone_Density_And_Structure
- Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias
- Comorb_Osteoporosis_without_current_pathological_fracture
- Comorb_Personal_history_of_malignant_neoplasm
- Comorb_Gastro_esophageal_reflux_disease
- Concom_Cholesterol_And_Triglyceride_Regulating_Preparations
- Concom_Narcotics
- Concom_Systemic_Corticosteroids_Plain
- Concom_Anti_Depressants_And_Mood_Stabilisers
- Concom_Fluoroquinolones
- Concom_Cephalosporins
- Concom_Macrolides_And_Similar_Types
- Concom_Broad_Spectrum_Penicillins
- Concom_Anaesthetics_General
- Concom_Viral_Vaccines
- Risk_Type_1_Insulin_Dependent_Diabetes
- Risk_Osteogenesis_Imperfecta
- Risk_Rheumatoid_Arthritis
- Risk_Untreated_Chronic_Hyperthyroidism
- Risk_Untreated_Chronic_Hypogonadism
- Risk_Untreated_Early_Menopause
- Risk_Patient_Parent_Fractured_Their_Hip
- Risk_Smoking_Tobacco
- Risk_Chronic_Malnutrition_Or_Malabsorption
- Risk_Chronic_Liver_Disease
- Risk_Family_History_Of_Osteoporosis

- Risk_Low_Calcium_Intake
- Risk_Vitamin_D_Insufficiency
- Risk_Poor_Health_Frailty
- Risk_Excessive_Thinness
- Risk_Hysterectomy_Oophorectomy
- Risk_Estrogen_Deficiency
- Risk Immobilization
- Risk_Recurring_Falls

4.2. Output variable (desired target)

- Persistency_Flag: Drug is persistent or not (Binary: Persistent, Non-Persistent)

4.3. Application Workflow

Given Workflow shows K- Nearest Neighbors Classifier model is used and Flask Framework for deployment. It represents the details of how the model works from user interface till the results.

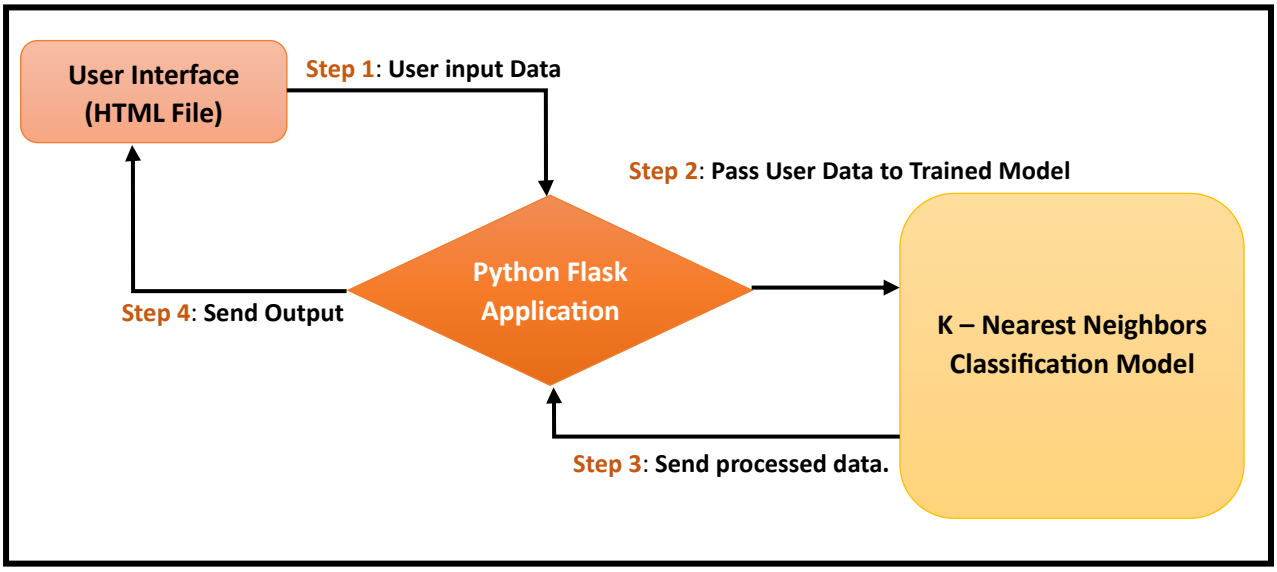


Fig 4.1 Application Framework

The machine learning model is built for Prediction of Persistency of the drug based on input attributes, then creates an API for the model using flask Framework and python micro-framework for building web application. This API call used to predict results through HTTP requests.

5. Building Machine Learning Model

5.1. Import Data Set

Import dataset for model training and building.

```
In [1]: import pandas as pd    # For data manupulation using dataframes
        import numpy as np     # For Statistical Analysis
        import math
```

```
In [2]: d=pd.read_excel('Healthcare_dataset.xlsx')
        d.head()
```

Out[2]:

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

5 rows × 69 columns

5.2. Dataset Details

- Shape of the dataset (Number of rows and columns)

```
In [3]: # no of rows and columns
        d.shape

Out[3]: (3424, 69)
```

Number of rows = 3424
Number of columns = 69

- Datatype of Columns and Non-null values

```
In [4]: # Datatypes of columns and non-null values
        d.info()

<class 'pandas.core.frame.DataFrame'>
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
 20  Change_T_Score                                        3424 non-null   object
 21  Change_Risk_Segment                                   3424 non-null   object
 22  Adherent_Flag                                         3424 non-null   object
 23  Idn_Indicator                                         3424 non-null   object
 24  Injectable_Experience_During_Rx                       3424 non-null   object
 25  Comorb_Encounter_For_Screening_For_Malignant_Neoplasms 3424 non-null  object
 26  Comorb_Encounter_For_Immunization                     3424 non-null   object
 27  Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx 3424 non-null object
 28  Comorb_Vitamin_D_Deficiency                           3424 non-null   object
 29  Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified  3424 non-null   object
 30  Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx 3424 non-null object
 31  Comorb_Long_Term_Current_Drug_Therapy                 3424 non-null   object
```

```
 32  Comorb_Dorsalgia                                          3424 non-null   object
 33  Comorb_Personal_History_Of_Other_Diseases_And_Conditions  3424 non-null   object
 34  Comorb_Other_Disorders_Of_Bone_Density_And_Structure      3424 non-null   object
 35  Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias  3424 non-null   object
 36  Comorb_Osteoporosis_without_current_pathological_fracture  3424 non-null   object
 37  Comorb_Personal_history_of_malignant_neoplasm             3424 non-null   object
 38  Comorb_Gastro_esophageal_reflux_disease                   3424 non-null   object
 39  Concom_Cholesterol_And_Triglyceride_Regulating_Preparations  3424 non-null   object
 40  Concom_Narcotics                                          3424 non-null   object
 41  Concom_Systemic_Corticosteroids_Plain                     3424 non-null   object
 42  Concom_Anti_Depressants_And_Mood_Stabilisers              3424 non-null   object
 43  Concom_Fluoroquinolones                                   3424 non-null   object
 44  Concom_Cephalosporins                                     3424 non-null   object
 45  Concom_Macrolides_And_Similar_Types                       3424 non-null   object
 46  Concom_Broad_Spectrum_Penicillins                         3424 non-null   object
 47  Concom_Anaesthetics_General                               3424 non-null   object
 48  Concom_Viral_Vaccines                                     3424 non-null   object
 49  Risk_Type_1_Insulin_Dependent_Diabetes                    3424 non-null   object
 50  Risk_Osteogenesis_Imperfecta                              3424 non-null   object
 51  Risk_Rheumatoid_Arthritis                                 3424 non-null   object
 52  Risk_Untreated_Chronic_Hyperthyroidism                    3424 non-null   object
 53  Risk_Untreated_Chronic_Hypogonadism                       3424 non-null   object
 54  Risk_Untreated_Early_Menopause                            3424 non-null   object
 55  Risk_Patient_Parent_Fractured_Their_Hip                   3424 non-null   object
 56  Risk_Smoking_Tobacco                                      3424 non-null   object
 57  Risk_Chronic_Malnutrition_Or_Malabsorption                3424 non-null   object
 58  Risk_Chronic_Liver_Disease                                3424 non-null   object
 59  Risk_Family_History_Of_Osteoporosis                       3424 non-null   object
 60  Risk_Low_Calcium_Intake                                   3424 non-null   object
 61  Risk_Vitamin_D_Insufficiency                              3424 non-null   object
 62  Risk_Poor_Health_Frailty                                  3424 non-null   object
 63  Risk_Excessive_Thinness                                   3424 non-null   object
 64  Risk_Hysterectomy_Oophorectomy                            3424 non-null   object
 65  Risk_Estrogen_Deficiency                                  3424 non-null   object
 66  Risk_Immobilization                                       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
```

- Numerical and categorical Features

```
In [5]: # Function to identify numeric features
        def numeric_features(dataset):
            numeric_col = dataset.select_dtypes(include=['number']).columns
            return numeric_col

        # Function to identify categorical features
        def categorical_features(dataset):
            categorical_col = dataset.select_dtypes(exclude=['number']).columns
            return categorical_col

In [6]: # display numeric and categorical features
        def display_numeric_categoric_feature(dataset):
            numeric_columns = numeric_features(dataset)
            print("Numeric Features:")
            print(numeric_columns)
            print("===="*20)
            categorical_columns = categorical_features(dataset)
            print("Categorical Features:")
            print(categorical_columns)
```

```
In [7]: display_numeric_categoric_feature(d)

        Numeric Features:
        Index(['Dexa_Freq_During_Rx', 'Count_Of_Risks'], dtype='object')
        ================================================================================
        Categorical Features:
        Index(['Ptid', 'Persistency_Flag', 'Gender', 'Race', 'Ethnicity', 'Region',
               'Age_Bucket', 'Ntm_Speciality', 'Ntm_Specialist_Flag',
               'Ntm_Speciality_Bucket', 'Gluco_Record_Prior_Ntm',
               'Gluco_Record_During_Rx', 'Dexa_During_Rx', 'Frag_Frac_Prior_Ntm',
               'Frag_Frac_During_Rx', 'Risk_Segment_Prior_Ntm',
               'Tscore_Bucket_Prior_Ntm', 'Risk_Segment_During_Rx',
               'Tscore_Bucket_During_Rx', 'Change_T_Score', 'Change_Risk_Segment',
               'Adherent_Flag', 'Idn_Indicator', 'Injectable_Experience_During_Rx',
               'Comorb_Encounter_For_Screening_For_Malignant_Neoplasms',
               'Comorb_Encounter_For_Immunization',
               'Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx',
               'Comorb_Vitamin_D_Deficiency',
               'Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified',
               'Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx',
               'Comorb_Long_Term_Current_Drug_Therapy', 'Comorb_Dorsalgia',
               'Comorb_Personal_History_Of_Other_Diseases_And_Conditions',
               'Comorb_Other_Disorders_Of_Bone_Density_And_Structure',
               'Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias',
               'Comorb_Osteoporosis_without_current_pathological_fracture',
               'Comorb_Personal_history_of_malignant_neoplasm',
               'Comorb_Gastro_esophageal_reflux_disease',
               'Concom_Cholesterol_And_Triglyceride_Regulating_Preparations',
               'Concom_Narcotics', 'Concom_Systemic_Corticosteroids_Plain',
               'Concom_Anti_Depressants_And_Mood_Stabilisers',
               'Concom_Fluoroquinolones', 'Concom_Cephalosporins',
               'Concom_Macrolides_And_Similar_Types',
               'Concom_Broad_Spectrum_Penicillins', 'Concom_Anaesthetics_General',
               'Concom_Viral_Vaccines', 'Risk_Type_1_Insulin_Dependent_Diabetes',
               'Risk_Osteogenesis_Imperfecta', 'Risk_Rheumatoid_Arthritis',
               'Risk_Untreated_Chronic_Hyperthyroidism',
               'Risk_Untreated_Chronic_Hypogonadism', 'Risk_Untreated_Early_Menopause',
               'Risk_Patient_Parent_Fractured_Their_Hip', 'Risk_Smoking_Tobacco',
               'Risk_Chronic_Malnutrition_Or_Malabsorption',
               'Risk_Chronic_Liver_Disease', 'Risk_Family_History_Of_Osteoporosis',
               'Risk_Low_Calcium_Intake', 'Risk_Vitamin_D_Insufficiency',
               'Risk_Poor_Health_Frailty', 'Risk_Excessive_Thinness',
               'Risk_Hysterectomy_Oophorectomy', 'Risk_Estrogen_Deficiency',
               'Risk_Immobilization', 'Risk_Recurring_Falls'],
              dtype='object')
```

- Null values

```
In [8]: # total null values in the dataset
        d.isnull().sum()

Out[8]: 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

In [9]: d.isna().apply(pd.value_counts)

Out[9]:
```

	Ptid	Persistency_Flag	Gender	Race	Ethnicity	Region	Age_Bucket	Ntm_Speciality	Ntm_Specialist_Flag	Ntm_Speciality_Bucket	...	Risk_Family_Histo
False	3424	3424	3424	3424	3424	3424	3424	3424	3424	3424	...	

1 rows × 69 columns

There are no null values in the dataset.

6. Dataset Pre-processing and Visualization

- Drop Duplicate rows.

```
In [12]: # Remove duplicate rows
         d=d.drop_duplicates()
         d
```

Out[12]:

	Ptid	Persistency_Flag	Gender	Race	Ethnicity	Region	Age_Bucket	Ntm_Speciality	Ntm_Specialist_Flag	Ntm_Speciality_Bucket	...
0	P1	Persistent	Male	Caucasian	Not Hispanic	West	>75	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	...
1	P2	Non-Persistent	Male	Asian	Not Hispanic	West	55-65	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	...
2	P3	Non-Persistent	Female	Other/Unknown	Hispanic	Midwest	65-75	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	...
3	P4	Non-Persistent	Female	Caucasian	Not Hispanic	Midwest	>75	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	...
4	P5	Non-Persistent	Female	Caucasian	Not Hispanic	Midwest	>75	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	...
...	...	...	...	...	...	...	...	...	...	...	...
3419	P3420	Persistent	Female	Caucasian	Not Hispanic	South	>75	GENERAL PRACTITIONER	Others	OB/GYN/Others/PCP/Unknown	...
3420	P3421	Persistent	Female	Caucasian	Not Hispanic	South	>75	Unknown	Others	OB/GYN/Others/PCP/Unknown	...
3421	P3422	Persistent	Female	Caucasian	Not Hispanic	South	>75	ENDOCRINOLOGY	Specialist	Endo/Onc/Uro	...
3422	P3423	Non-Persistent	Female	Caucasian	Not Hispanic	South	55-65	Unknown	Others	OB/GYN/Others/PCP/Unknown	...
3423	P3424	Non-Persistent	Female	Caucasian	Not Hispanic	South	65-75	Unknown	Others	OB/GYN/Others/PCP/Unknown	...

3424 rows × 69 columns

There are no duplicate rows.

- Drop unnecessary columns.

```
In [13]: #Drop ptid column
         d.drop(columns='Ptid',axis=1,inplace=True)
```

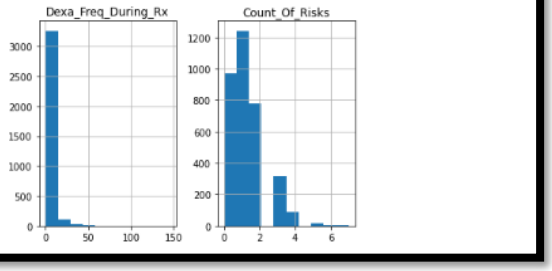
'Ptid' has all unique values that's why it has dropped.

6.1. Univariate Analysis for Continuous Columns

```
In [14]: import matplotlib.pyplot as plt       # For Data Visualisation
         import seaborn as sns                 # for statistical Data Visualisation
         import warnings
         warnings.filterwarnings('ignore')
```

- Histogram for continuous columns

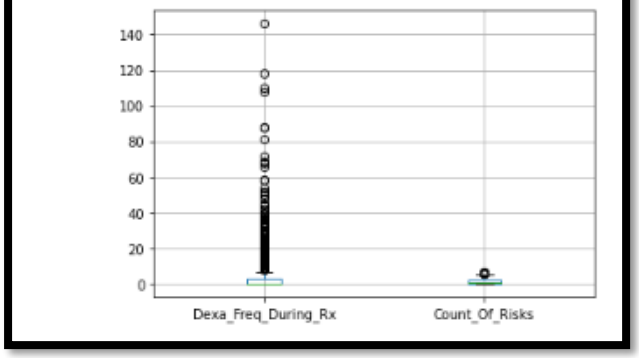
```
In [15]: d.hist()
Out[15]: array([[<AxesSubplot:title={'center':'Dexa_Freq_During_Rx'}>,
                 <AxesSubplot:title={'center':'Count_Of_Risks'}>]], dtype=object)
```



Both numerical attributes have un-even distribution.

- Boxplot for continuous columns

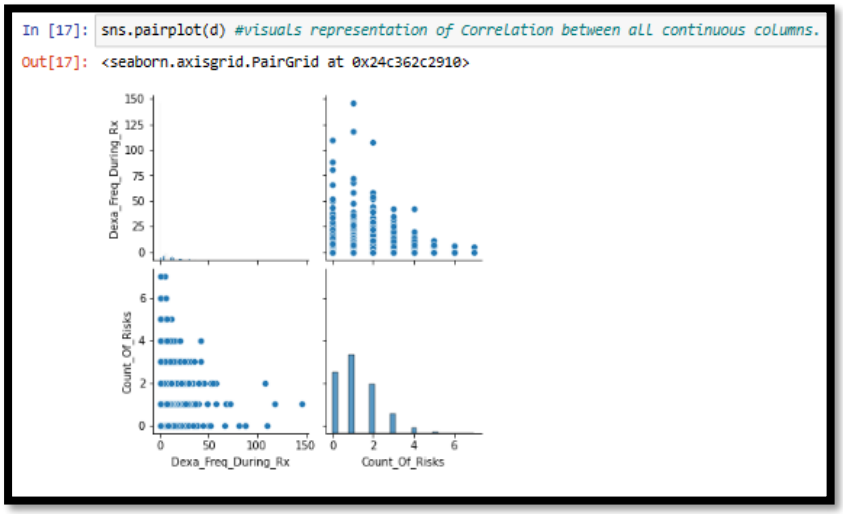
```
In [16]: d.boxplot()
Out[16]: <AxesSubplot:>
```



Both numerical variables have outliers.

- Pairplot for Numerical columns

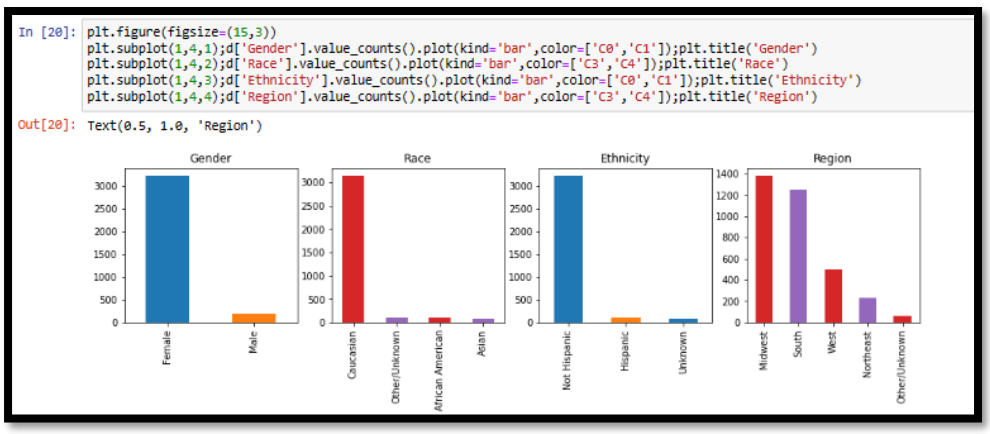
```
In [17]: sns.pairplot(d) #visuals representation of Correlation between all continuous columns.
Out[17]: <seaborn.axisgrid.PairGrid at 0x24c362c2910>
```



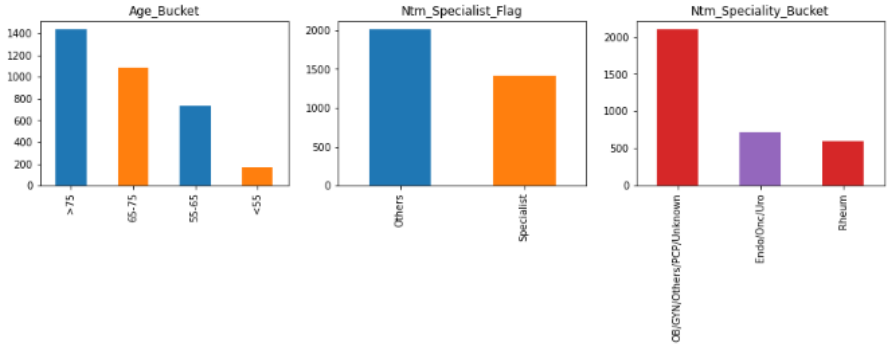
There is very less correlation between the numerical variables.

6.2. Univariate Analysis for Categorical Variables

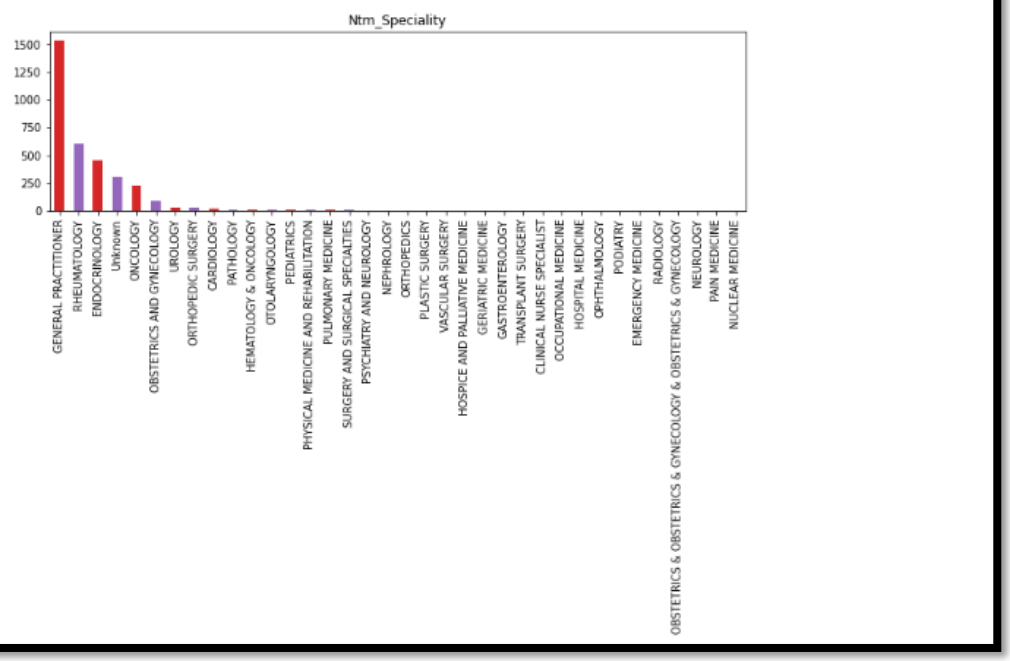
```
In [20]: plt.figure(figsize=(15,3))
         plt.subplot(1,4,1);d['Gender'].value_counts().plot(kind='bar',color=['C0','C1']);plt.title('Gender')
         plt.subplot(1,4,2);d['Race'].value_counts().plot(kind='bar',color=['C3','C4']);plt.title('Race')
         plt.subplot(1,4,3);d['Ethnicity'].value_counts().plot(kind='bar',color=['C0','C1']);plt.title('Ethnicity')
         plt.subplot(1,4,4);d['Region'].value_counts().plot(kind='bar',color=['C3','C4']);plt.title('Region')
Out[20]: Text(0.5, 1.0, 'Region')
```



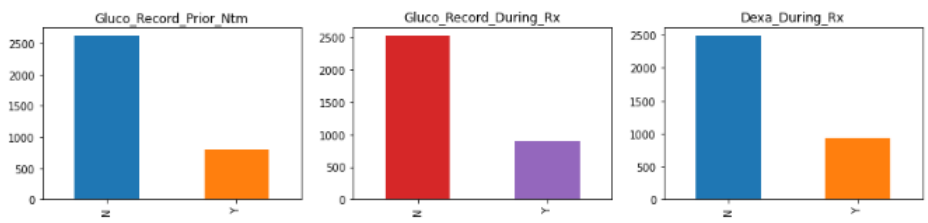
```
In [24]: plt.figure(figsize=(15,3))
         plt.subplot(1,3,1);d['Age_Bucket'].value_counts().plot(kind='bar',color=['C0','C1']);plt.title('Age_Bucket')
         plt.subplot(1,3,2);d['Ntm_Specialist_Flag'].value_counts().plot(kind='bar',color=['C0','C1']);plt.title('Ntm_Specialist_Flag')
         plt.subplot(1,3,3);d['Ntm_Speciality_Bucket'].value_counts().plot(kind='bar',color=['C3','C4']);plt.title('Ntm_Speciality_Bucket
Out[24]: Text(0.5, 1.0, 'Ntm_Speciality_Bucket')
```




```
In [25]: plt.figure(figsize=(25,3))
         plt.subplot(1,2,2);d['Ntm_Speciality'].value_counts().plot(kind='bar',color=['C3','C4']);plt.title('Ntm_Speciality')
Out[25]: Text(0.5, 1.0, 'Ntm_Speciality')
```

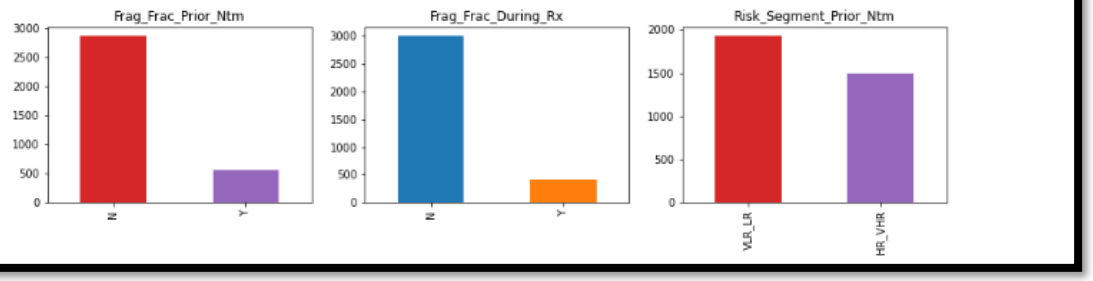



```
In [26]: plt.figure(figsize=(15,3))
         plt.subplot(1,3,1);d['Gluco_Record_Prior_Ntm'].value_counts().plot(kind='bar',color=['C0','C1']);plt.title('Gluco_Record_Prior_N
         plt.subplot(1,3,2);d['Gluco_Record_During_Rx'].value_counts().plot(kind='bar',color=['C3','C4']);plt.title('Gluco_Record_During_
         plt.subplot(1,3,3);d['Dexa_During_Rx'].value_counts().plot(kind='bar',color=['C0','C1']);plt.title('Dexa_During_Rx')
Out[26]: Text(0.5, 1.0, 'Dexa_During_Rx')
```



```
In [27]: plt.figure(figsize=(15,3))
         plt.subplot(1,3,1);d['Frag_Frac_Prior_Ntm'].value_counts().plot(kind='bar',color=['C3','C4']);plt.title('Frag_Frac_Prior_Ntm')
         plt.subplot(1,3,2);d['Frag_Frac_During_Rx'].value_counts().plot(kind='bar',color=['C0','C1']);plt.title('Frag_Frac_During_Rx')
         plt.subplot(1,3,3);d['Risk_Segment_Prior_Ntm'].value_counts().plot(kind='bar',color=['C3','C4']);plt.title('Risk_Segment_Prior_N

Out[27]: Text(0.5, 1.0, 'Risk_Segment_Prior_Ntm')
```

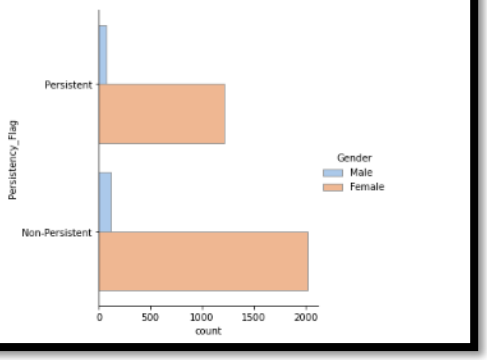


Most of the categorical variables have uneven distribution.

6.3. Bivariate Analysis

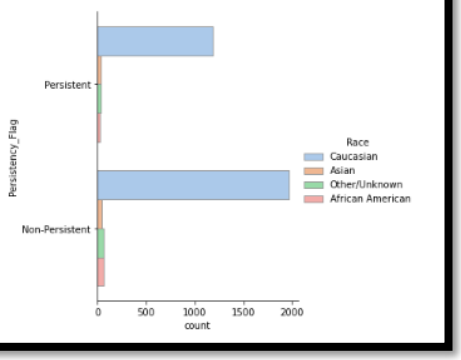
```
In [28]: sns.catplot(
             data=d, y="Persistency_Flag", hue="Gender", kind="count",
             palette="pastel", edgecolor=".6",)

Out[28]: <seaborn.axisgrid.FacetGrid at 0x24c3ee3cb50>
```



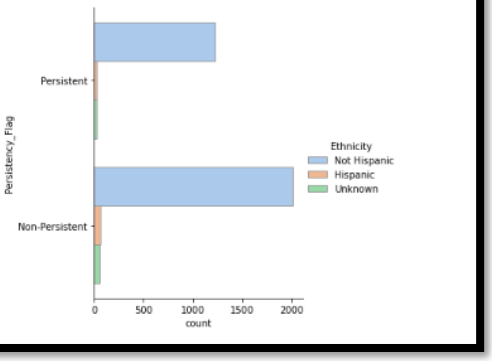

```
In [29]: sns.catplot(
             data=d, y="Persistency_Flag", hue="Race", kind="count",
             palette="pastel", edgecolor=".6",)

Out[29]: <seaborn.axisgrid.FacetGrid at 0x24c3ee47370>
```



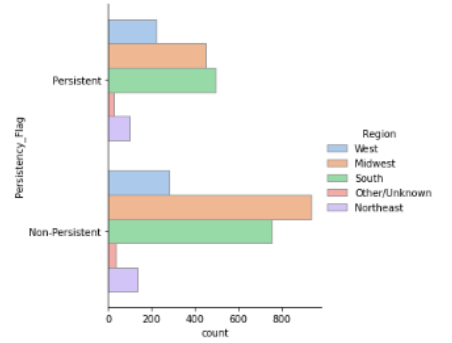

```
In [30]: sns.catplot(
             data=d, y="Persistency_Flag", hue="Ethnicity", kind="count",
             palette="pastel", edgecolor=".6",)

Out[30]: <seaborn.axisgrid.FacetGrid at 0x24c3eb63af0>
```

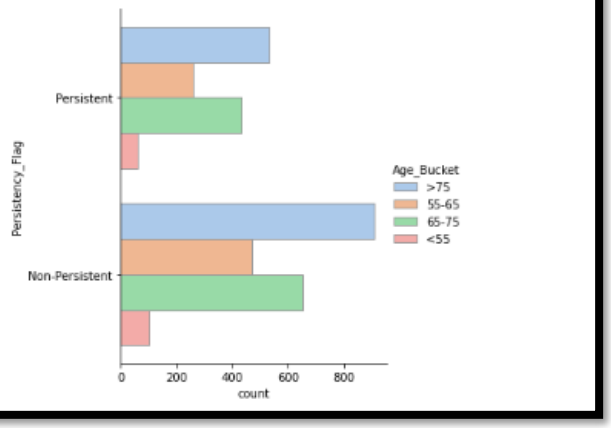



```
In [31]: sns.catplot(
             data=d, y="Persistency_Flag", hue="Region", kind="count",
             palette="pastel", edgecolor=".6",)

Out[31]: <seaborn.axisgrid.FacetGrid at 0x24c3ff10460>
```



```
In [32]: sns.catplot(
             data=d, y="Persistency_Flag", hue="Age_Bucket", kind="count",
             palette="pastel", edgecolor=".6",)
Out[32]: <seaborn.axisgrid.FacetGrid at 0x24c4007a490>
```



There is no significant insight through bivariate Analysis.

7. Categorical Features Selection

- Select categorical features that are stored in variable “categorical_columns”.

```
In [34]: categorical_columns = categorical_features(d)
```

- Convert datatype from object to category.

```
In [35]: # Convert object to category
         d[categorical_columns]=d[categorical_columns].astype("category")
```

- Encode categorical features into numeric.

```
In [37]: # encoding categorical features into numeric
         d[categorical_columns]=d[categorical_columns].apply(lambda x: x.cat.codes)
```

- Divide Categorical columns into input and target variables.

```
In [39]: x=d[categorical_columns].drop(columns=['Persistency_Flag'])
         y=d['Persistency_Flag']
```

7.1. Categorical Feature Selection using sklearn library and chi2 and SelectKbest function.

```
In [40]: from sklearn.feature_selection import chi2, SelectKBest

In [41]: cs= SelectKBest (score_func = chi2, k= "all")
         cs.fit(x,y)
         feature_score = pd.DataFrame({"Score":cs.scores_, "P_Values": cs.pvalues_},index = x.columns)
         feature_score.nlargest(n=61, columns="Score")
```

Out[41]:

	Score	P_Values
Dexa_During_Rx	601.821735	6.722923e-133
Comorb_Long_Term_Current_Drug_Therapy	324.413431	1.583364e-72
Comorb_Encounter_For_Screening_For_Malignant_Neoplasms	196.456669	1.239081e-44
Comorb_Encounter_For_Immunization	189.482869	4.122973e-43
Comorb_Other_Disorders_Of_Bone_Density_And_Structure	177.698458	1.541551e-40
...	...	...
Risk_Untreated_Early_Menopause	0.095081	7.578140e-01
Gluco_Record_Prior_Ntm	0.086825	7.682533e-01
Risk_Family_History_Of_Osteoporosis	0.037398	8.466578e-01
Risk_Osteogenesis_Imperfecta	0.023770	8.774707e-01
Age_Bucket	0.011883	9.131938e-01

61 rows × 2 columns

- Eliminate Categorical features with less or no relationship with target variable considering the p-value> 0.05 .

```
In [42]: #drop categoricalcolumns with less or no relationship with traget varibale
         remove_columns = ['Ntm_Speciality','Gender','Risk_Low_Calcium_Intake','Risk_Segment_Prior_Ntm',
                           'Risk_Patient_Parent_Fractured_Their_Hip','Change_Risk_Segment','Risk_Untreated_Early_Menopause',
                           'Gluco_Record_Prior_Ntm','Risk_Family_History_Of_Osteoporosis','Risk_Osteogenesis_Imperfecta','Age_Bucket',
                           'Race','Risk_Segment_During_Rx','Ethnicity','Frag_Frac_Prior_Ntm']

         d.drop(columns=remove_columns,inplace=True)
```

- Final dataset ready for Modelling

```
In [43]: d.sample(10)
```

Out[43]:

	Persistency_Flag	Region	Ntm_Specialist_Flag	Ntm_Speciality_Bucket	Gluco_Record_During_Rx	Dexa_Freq_During_Rx	Dexa_During_Rx	Frag_Frac_Durin
2632	1	0	0	1	0	0	0	
908	1	3	1	2	0	0	0	
723	0	0	0	1	0	0	0	
138	0	0	0	1	0	0	0	
2077	0	4	0	1	0	0	0	
2008	0	3	0	1	0	0	0	
1780	1	4	1	1	0	0	0	
1624	1	2	0	1	0	0	0	
3332	0	1	1	0	0	0	0	
3013	1	0	1	2	1	8	1	

10 rows × 53 columns

```
In [44]: d.shape
Out[44]: (3424, 53)

In [45]: # List of columns
         d.columns
Out[45]: Index(['Persistency_Flag', 'Region', 'Ntm_Specialist_Flag',
                'Ntm_Speciality_Bucket', 'Gluco_Record_During_Rx',
                'Dexa_Freq_During_Rx', 'Dexa_During_Rx', 'Frag_Frac_During_Rx',
                'Tscore_Bucket_Prior_Ntm', 'Tscore_Bucket_During_Rx', 'Change_T_Score',
                'Adherent_Flag', 'Idn_Indicator', 'Injectable_Experience_During_Rx',
                'Comorb_Encounter_For_Screening_For_Malignant_Neoplasms',
                'Comorb_Encounter_For_Immunization',
                'Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx',
                'Comorb_Vitamin_D_Deficiency',
                'Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified',
                'Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx',
                'Comorb_Long_Term_Current_Drug_Therapy', 'Comorb_Dorsalgia',
                'Comorb_Personal_History_Of_Other_Diseases_And_Conditions',
                'Comorb_Other_Disorders_Of_Bone_Density_And_Structure',
                'Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias',
                'Comorb_Osteoporosis_without_current_pathological_fracture',
                'Comorb_Personal_history_of_malignant_neoplasm',
                'Comorb_Gastro_esophageal_reflux_disease',
                'Concom_Cholesterol_And_Triglyceride_Regulating_Preparations',
                'Concom_Narcotics', 'Concom_Systemic_Corticosteroids_Plain',
                'Concom_Anti_Depressants_And_Mood_Stabilisers',
                'Concom_Fluoroquinolones', 'Concom_Cephalosporins',
                'Concom_Macrolides_And_Similar_Types',
                'Concom_Broad_Spectrum_Penicillins', 'Concom_Anaesthetics_General',
                'Concom_Viral_Vaccines', 'Risk_Type_1_Insulin_Dependent_Diabetes',
                'Risk_Rheumatoid_Arthritis', 'Risk_Untreated_Chronic_Hyperthyroidism',
                'Risk_Untreated_Chronic_Hypogonadism', 'Risk_Smoking_Tobacco',
                'Risk_Chronic_Malnutrition_Or_Malabsorption',
                'Risk_Chronic_Liver_Disease', 'Risk_Vitamin_D_Insufficiency',
                'Risk_Poor_Health_Frailty', 'Risk_Excessive_Thinness',
                'Risk_Hysterectomy_Oophorectomy', 'Risk_Estrogen_Deficiency',
                'Risk_Immobilization', 'Risk_Recurring_Falls', 'Count_Of_Risks'],
               dtype='object')
```

8. Model Building and Model Selection

8.1. Balance the dataset

```
In [47]: X= d.drop(columns='Persistency_Flag')
         Y = d['Persistency_Flag']
```

```
In [48]: Y.value_counts()
Out[48]: 0    2135
         1    1289
         Name: Persistency_Flag, dtype: int64
```

The dataset is imbalance, so we will balance the dataset using SMOTE.

```
In [49]: # balance the dataset using SMOTE
         from imblearn.over_sampling import SMOTE
         from collections import Counter

In [50]: sm = SMOTE(random_state=42)
         X_res, Y_res = sm.fit_resample(X, Y)
         print('Resampled dataset shape %s' % Counter(Y_res))

         Resampled dataset shape Counter({1: 2135, 0: 2135})

In [51]: Y_res.value_counts()
Out[51]: 1    2135
         0    2135
         Name: Persistency_Flag, dtype: int64
```

8.2. Split dataset into Train and Test datasets

Import train_test_split and divide the dataset into input variables and output variable then split the input and output into train and test sets (20% test and 80% train).

```
from sklearn.model_selection import train_test_split
```

```
In [52]: # splitting dataset in 80% train dataset and 20% test dataset
         X_train,X_test,Y_train,Y_test = train_test_split(X_res,Y_res, test_size=0.2,random_state=42)

In [53]: X_train.shape
Out[53]: (3416, 52)

In [54]: X_test.shape
Out[54]: (854, 52)
```

8.3. Logistic Regression Model

After data pre-processing, a machine learning model is created to predict the persistency of the drug. For this purpose, Logistic regression algorithm is used from sklearn. linear_model. After importing and initialize Logistic Regression model the dataset is being fitted for training using classifier.

```
In [55]: from sklearn.linear_model import LogisticRegression

In [56]: reg=LogisticRegression()
         reg.fit(X_train,Y_train) # Fit the model to the training data
Out[56]: LogisticRegression()

In [57]: Y_pred=reg.predict(X_test) # Predict the classes on the test data
         Y_pred
         ...

In [58]: np.mean(Y_pred==Y_test)
Out[58]: 0.7740046838407494

In [59]: pd.crosstab(Y_test,Y_pred)
Out[59]:
col_0              0    1
Persistency_Flag
0                335   83
1                110  326

In [60]: lreg_data=reg.score(X,Y)
         lreg_train=reg.score(X_train,Y_train)
         lreg_test=reg.score(X_test,Y_test)
         print ("Accuracy of All dataset: " ,(lreg_data))
         print ("Accuracy of Train dataset: " ,(lreg_train))
         print ("Accuracy of Test dataset: " ,(lreg_test))

         Accuracy of All dataset:  0.8022780373831776
         Accuracy of Train dataset:  0.7854215456674473
         Accuracy of Test dataset:  0.7740046838407494
```

The score of the Logistic Regression model is fine. Let's train the model with another algorithm to find better model than Logistic Regression Model.

8.4. Random Forest Classifier

Random Forest classification algorithm is used from sklearn. ensemble. After importing and initialize Random Forest classification model the dataset is being fitted for training using clf.

```
In [61]: from sklearn.ensemble import RandomForestClassifier

In [62]: clf = RandomForestClassifier(max_depth=3, random_state=42)
         clf.fit(X_train,Y_train) # Fit the model to the training data

Out[62]: RandomForestClassifier(max_depth=3, random_state=42)

In [63]: Y1_pred=clf.predict(X_test) # Predict the classes on the test data
         Y1_pred

         ...

In [64]: np.mean(Y1_pred==Y_test)

Out[64]: 0.7517564402810304

In [65]: pd.crosstab(Y_test,Y1_pred)

Out[65]:
col_0              0    1
Persistency_Flag
0                369   49
1                163  273

In [66]: rft_data=clf.score(X,Y)
         rft_train=clf.score(X_train,Y_train)
         rft_test=clf.score(X_test,Y_test)
         print ("Accuracy of All dataset: " ,(rft_data))
         print ("Accuracy of Train dataset: " ,(rft_train))
         print ("Accuracy of Test dataset: " ,(rft_test))

         Accuracy of All dataset:  0.8028621495327103
         Accuracy of Train dataset:  0.7681498829039812
         Accuracy of Test dataset:  0.7517564402810304
```

The Random Forest classification model score is lower than the Logistic Regression model but there are less false negatives now we try another algorithm to train the model that is K Nearest Neighbor Classifier.

8.5. K – Nearest Neighbor Classifier

K- Nearest Neighbor classification algorithm is used from sklearn. Neighbor. After importing and initialize KNNC model the dataset is being fitted for training using neigh.

```
In [67]: from sklearn.neighbors import KNeighborsClassifier

In [68]: neigh = KNeighborsClassifier(n_neighbors=5)
         neigh.fit(X_train,Y_train) # Fit the model to the training data

Out[68]: KNeighborsClassifier()

In [69]: YK_pred=neigh.predict(X_test) # Predict the classes on the test data
         YK_pred

                                                                    ...

In [70]: np.mean(YK_pred==Y_test)

Out[70]: 0.7868852459016393

In [71]: pd.crosstab(Y_test,YK_pred)

Out[71]:
col_0              0    1
Persistency_Flag
                 0 332   86
                 1  96  340

In [72]: knc_data=neigh.score(X,Y)
         knc_train=neigh.score(X_train,Y_train)
         knc_test=neigh.score(X_test,Y_test)
         print ("Accuracy of All dataset: " ,(knc_data))
         print ("Accuracy of Train dataset: " ,(knc_train))
         print ("Accuracy of Test dataset: " ,(knc_test))

         Accuracy of All dataset:  0.8373247663551402
         Accuracy of Train dataset:  0.8635831381733021
         Accuracy of Test dataset:  0.7868852459016393
```

The score of KNNC model for complete dataset is better. Let's try another algorithm to train the model that is Gradient Boosting Classifier.

8.6. Gradient Boosting Classifier

Gradient Boosting classification algorithm is used from sklearn. ensemble. After importing and initialize Gradient Boosting classification model the dataset is being fitted for training using model.

```
In [73]: from sklearn.ensemble import GradientBoostingClassifier

In [74]: model=GradientBoostingClassifier(n_estimators=300, learning_rate=1.0, max_depth=2, random_state=40)
         model.fit(X_train,Y_train) # Fit the model to the training data

Out[74]: GradientBoostingClassifier(learning_rate=1.0, max_depth=2, n_estimators=300,
                                    random_state=40)

In [75]: Y2_pred=model.predict(X_test) # Predict the classes on the test data
         Y2_pred

         ...

In [76]: np.mean(Y2_pred==Y_test)

Out[76]: 0.7704918032786885

In [77]: pd.crosstab(Y_test,Y2_pred)

Out[77]:
col_0              0    1
Persistency_Flag
0                333   85
1                111  325

In [78]: gbc_data=model.score(X,Y)
         gbc_train=model.score(X_train,Y_train)
         gbc_test=model.score(X_test,Y_test)
         print ("Accuracy of All dataset: " ,(gbc_data))
         print ("Accuracy of Train dataset: " ,(gbc_train))
         print ("Accuracy of Test dataset: " ,(gbc_test))

         Accuracy of All dataset:  0.8574766355140186
         Accuracy of Train dataset:  0.8679742388758782
         Accuracy of Test dataset:  0.7704918032786885
```

The score of Gradient Boosting classification model for complete dataset is best of all but it is also overfitting model. Next step is to perform hyperparameter tuning to try to improve the model.

8.7. Hyper-parameter Tuning

Accuracy of KNNC model and Gradient Boosting Classifier model are better than other models. Let's try to improve the accuracy of both the models, also try to fix the overfitting of Gradient Boosting Classification model. So, we will do hyperparameter tunning of these models. For hyperparameter tunning, Grid CV search from sklearn. model_selection will be used.

```
In [79]: from sklearn.model_selection import GridSearchCV
```

8.7.1. Hyperparameter tunning of KNNC Model

```
In [79]: from sklearn.model_selection import GridSearchCV

In [83]: #List Hyperparameters that we want to tune.
         leaf_size = list(range(1,20))
         n_neighbors = list(range(1,10))
         p=[1,2]
         #Convert to dictionary
         hyperparameters = dict(leaf_size=leaf_size, n_neighbors=n_neighbors, p=p)
         #Create new KNN object
         knn_2 = KNeighborsClassifier()

In [84]: #Use GridSearch
         modelK = GridSearchCV(knn_2, hyperparameters, cv=5)

In [85]: #Fit the model
         best_model = modelK.fit(X_train,Y_train)
         #Print The value of best Hyperparameters
         print('Best leaf_size:', best_model.best_estimator_.get_params()['leaf_size'])
         print('Best p:', best_model.best_estimator_.get_params()['p'])
         print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbors'])

         Best leaf_size: 1
         Best p: 2
         Best n_neighbors: 5
```

Now train the model using best hyperparameters.

```
In [86]: modelkn= KNeighborsClassifier(n_neighbors=5,p=2,leaf_size=1)

In [87]: modelkn.fit(X_train,Y_train)
Out[87]: KNeighborsClassifier(leaf_size=1)

In [88]: Ykn_pred=modelkn.predict(X_test)
         Ykn_pred

         ...

In [89]: np.mean(Ykn_pred==Y_test)
Out[89]: 0.7868852459016393

In [90]: pd.crosstab(Y_test,Ykn_pred)
Out[90]:
```

col_0	0	1
Persistency_Flag		
0	332	86
1	96	340

```
In [91]: knnc_data=modelkn.score(X,Y)
         knnc_train=modelkn.score(X_train,Y_train)
         knnc_test=modelkn.score(X_test,Y_test)
         print ("Accuracy of All dataset: " ,(knnc_data))
         print ("Accuracy of Train dataset: " ,(knnc_train))
         print ("Accuracy of Test dataset: " ,(knnc_test))

         Accuracy of All dataset:  0.8373247663551402
         Accuracy of Train dataset:  0.8635831381733021
         Accuracy of Test dataset:  0.7868852459016393
```

After Hyperparameter tuning of KNNC there is no improvement in the model. Let's do hyperparameter tuning of Gradient boosting Classifier model.

8.7.2. Hyperparameter tunning of Gradient Boosting Classification Model

```
In [92]: gb = GradientBoostingClassifier()
         parameters = {
             "n_estimators":[5,50,75,100],
             "max_depth":[1,3,5,7],
             "learning_rate":[0.01,0.1,1,10]}

In [93]: cv = GridSearchCV(gb,parameters,cv=5) # Here we are using 5 iterations
         cv.fit(X_train,Y_train)

Out[93]: GridSearchCV(cv=5, estimator=GradientBoostingClassifier(),
                      param_grid={'learning_rate': [0.01, 0.1, 1, 10],
                                  'max_depth': [1, 3, 5, 7],
                                  'n_estimators': [5, 50, 75, 100]})

In [94]: cv.best_params_

Out[94]: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 75}

In [139]: cv.best_score_

Out[139]: 0.8170374080638394
```

Now train the model using best hyperparameters.

```
In [95]: model1=GradientBoostingClassifier(n_estimators=75, learning_rate=0.1, max_depth=7, random_state=42)
         model1.fit(X_train,Y_train)

Out[95]: GradientBoostingClassifier(max_depth=7, n_estimators=75, random_state=42)

In [96]: YY_pred=model1.predict(X_test)
         YY_pred

...

In [97]: np.mean(YY_pred==Y_test)

Out[97]: 0.8032786885245902

In [98]: pd.crosstab(Y_test,YY_pred)

Out[98]:
col_0              0    1
Persistency_Flag
0                343   75
1                 93  343

In [99]: gb_data=model1.score(X,Y)
         gb_train=model1.score(X_train,Y_train)
         gb_test=model1.score(X_test,Y_test)
         print ("Accuracy of All dataset: " ,(gb_data))
         print ("Accuracy of Train dataset: " ,(gb_train))
         print ("Accuracy of Test dataset: " ,(gb_test))

         Accuracy of All dataset:  0.9427570093457944
         Accuracy of Train dataset:  0.9622365339578455
         Accuracy of Test dataset:  0.8032786885245902
```

After Hyperparameter tuning of Gradient Boosting model is still overfitting model.

9. Metrics for Evaluation

9.1. Accuracy, Precision, Recall and F1-Score

```
In [100]: from sklearn.metrics import classification_report, confusion_matrix
```

- Logistic Regression model

```
In [101]: #LogisticRegression
          print(classification_report(Y_test,Y_pred))
```

	precision	recall	f1-score	support
0	0.75	0.80	0.78	418
1	0.80	0.75	0.77	436
accuracy			0.77	854
macro avg	0.77	0.77	0.77	854
weighted avg	0.78	0.77	0.77	854

```
In [102]: confusion_matrix(Y_test,Y_pred)
Out[102]: array([[335,  83],
                 [110, 326]], dtype=int64)
```

- Random Forest Classification model

```
In [147]: #RandomForestTreeClassifier
          print(classification_report(Y_test,Y1_pred))
```

	precision	recall	f1-score	support
0	0.69	0.88	0.78	418
1	0.85	0.63	0.72	436
accuracy			0.75	854
macro avg	0.77	0.75	0.75	854
weighted avg	0.77	0.75	0.75	854

```
In [103]: confusion_matrix(Y_test,Y1_pred)
Out[103]: array([[369,  49],
                 [163, 273]], dtype=int64)
```

- K- Nearest Neighbor Classification model

```
In [166]: #KNeighborsClassifier without hyperparameter tuning
          print(classification_report(Y_test,YK_pred))
```

	precision	recall	f1-score	support
0	0.78	0.79	0.78	418
1	0.80	0.78	0.79	436
accuracy			0.79	854
macro avg	0.79	0.79	0.79	854
weighted avg	0.79	0.79	0.79	854

```
In [104]: confusion_matrix(Y_test,YK_pred)
Out[104]: array([[332,  86],
                 [ 96, 340]], dtype=int64)
```

- K- Nearest Neighbor Classification model with Hyperparameter Tuning

```
In [167]: #KNeighborsClassifier with hyperparameter tuning
          print(classification_report(Y_test,Ykn_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.79      0.78       418
           1       0.80      0.78      0.79       436

    accuracy                           0.79       854
   macro avg       0.79      0.79      0.79       854
weighted avg       0.79      0.79      0.79       854
```

```
In [105]: confusion_matrix(Y_test,Ykn_pred)
```

```
Out[105]: array([[332,  86],
                 [ 96, 340]], dtype=int64)
```

- Gradient Boosting Classification model

```
In [168]: #GradientBoostingClassifier withouthyper parameter tuning
          print(classification_report(Y_test,Y2_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.80      0.77       418
           1       0.79      0.75      0.77       436

    accuracy                           0.77       854
   macro avg       0.77      0.77      0.77       854
weighted avg       0.77      0.77      0.77       854
```

```
In [106]: confusion_matrix(Y_test,Y2_pred)
```

```
Out[106]: array([[333,  85],
                 [111, 325]], dtype=int64)
```

- Gradient Boosting Classification model with Hyperparameter Tuning

```
In [169]: #GradientBoostingClassifier with parameter tuning
          print(classification_report(Y_test,YY_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.76      0.78       418
           1       0.78      0.82      0.80       436

    accuracy                           0.79       854
   macro avg       0.79      0.79      0.79       854
weighted avg       0.79      0.79      0.79       854
```

```
In [107]: confusion_matrix(Y_test,YY_pred)
```

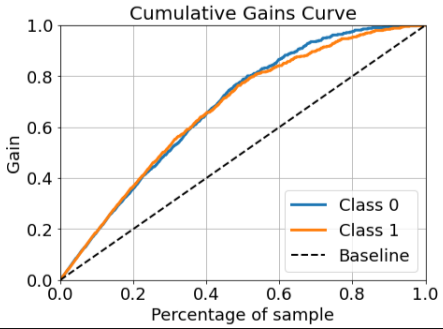
```
Out[107]: array([[343,  75],
                 [ 93, 343]], dtype=int64)
```

9.2. Lift and Gain

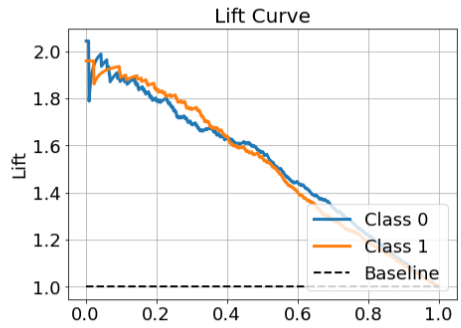
```
In [108]: import scikitplot as skplt
```

- Logistic Regression Model

```
In [109]: # Predict the classes on the test data, and return the probabilities for each class
Y_proba = reg.predict_proba(X_test)
skplt.metrics.plot_cumulative_gain(Y_test, Y_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
plt.show()
```

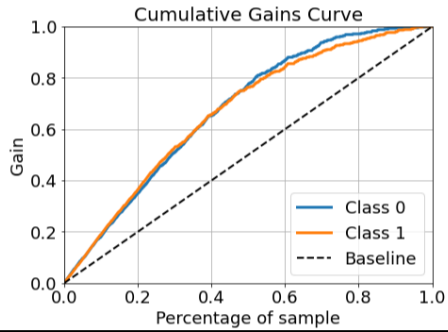



```
In [110]: skplt.metrics.plot_lift_curve(Y_test, Y_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
plt.show()
```

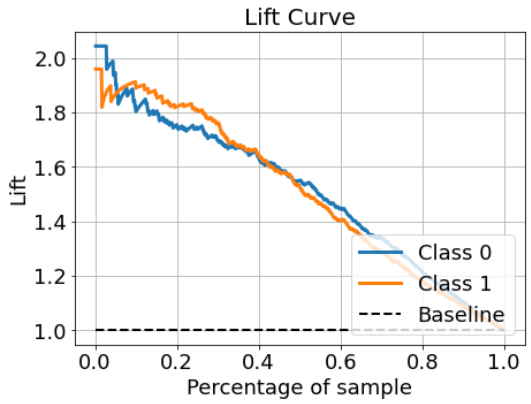


- Random Forest Classification Model

```
In [111]: # Predict the classes on the test data, and return the probabilities for each class
Y1_proba = clf.predict_proba(X_test)
skplt.metrics.plot_cumulative_gain(Y_test, Y1_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
plt.show()
```

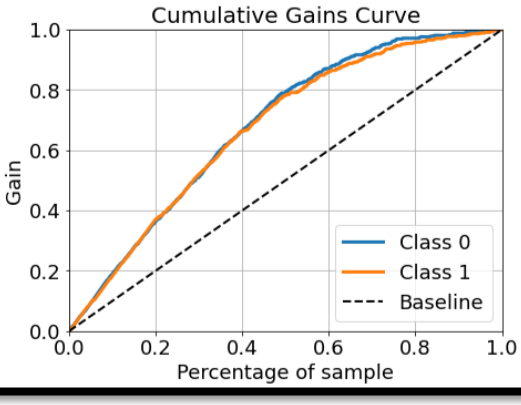


```
In [112]: skplt.metrics.plot_lift_curve(Y_test, Y1_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
          plt.show()
```

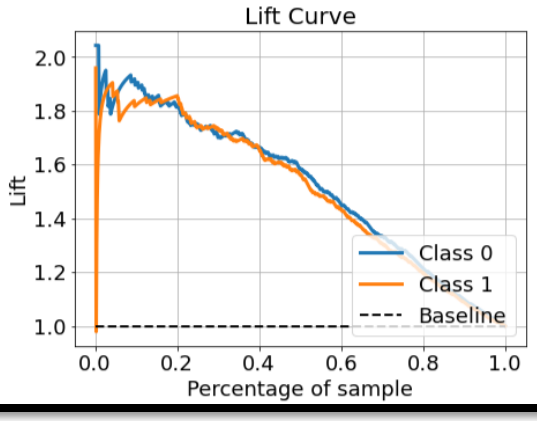


- K-Nearest Neighbor Classification Model

```
In [122]: # Predict the classes on the test data, and return the probabilities for each class
          YK_proba = neigh.predict_proba(X_test)
          skplt.metrics.plot_cumulative_gain(Y_test, YK_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
          plt.show()
```

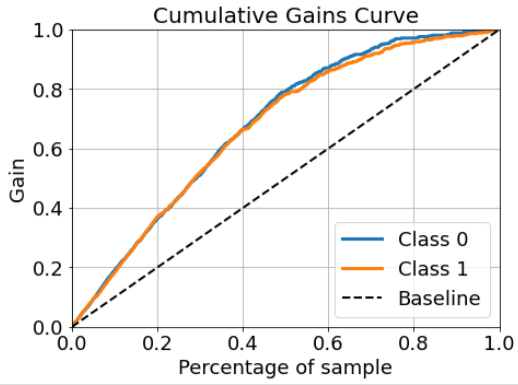



```
In [123]: skplt.metrics.plot_lift_curve(Y_test, YK_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
          plt.show()
```

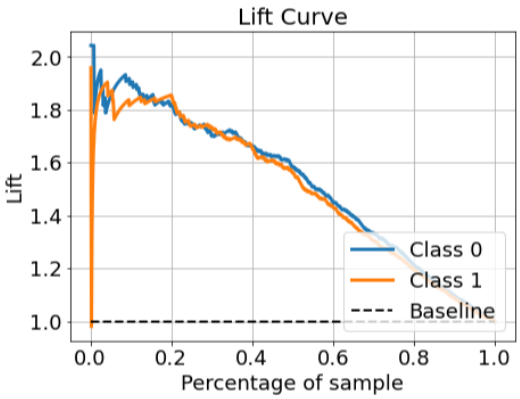


➢ K-Nearest Neighbor Classification Model with Hyperparameter Tuning

```
In [124]: # Predict the classes on the test data, and return the probabilities for each class
          Ykn_proba = modelkn.predict_proba(X_test)
          skplt.metrics.plot_cumulative_gain(Y_test, Ykn_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
          plt.show()
```

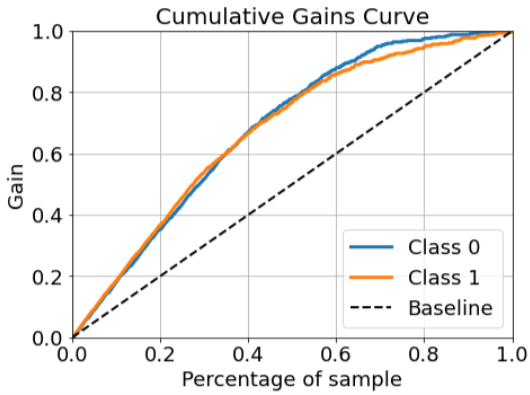



```
In [125]: skplt.metrics.plot_lift_curve(Y_test, Ykn_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
          plt.show()
```

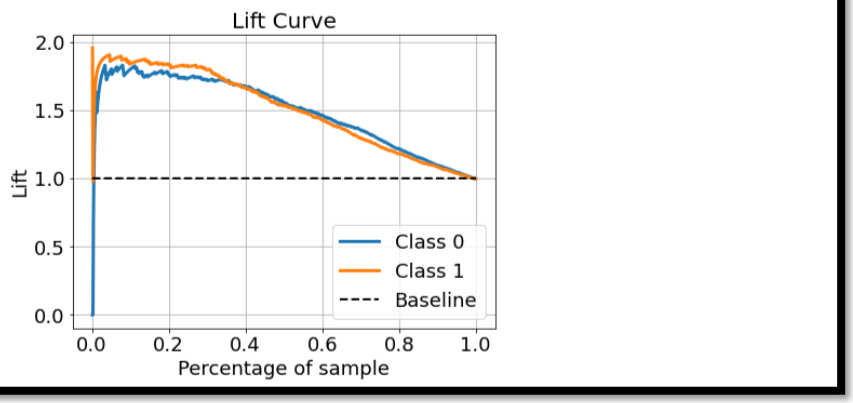


➢ Gradient Boosting Classification Model

```
In [126]: # Predict the classes on the test data, and return the probabilities for each class
          Y2_proba = model.predict_proba(X_test)
          skplt.metrics.plot_cumulative_gain(Y_test, Y2_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
          plt.show()
```

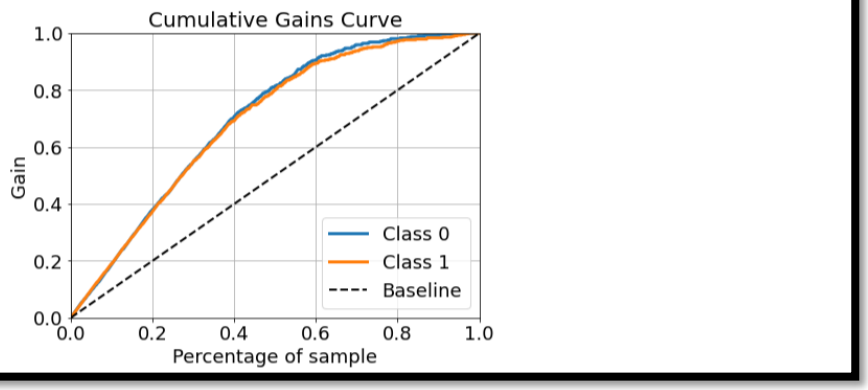


```
In [127]: skplt.metrics.plot_lift_curve(Y_test, Y2_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
          plt.show()
```

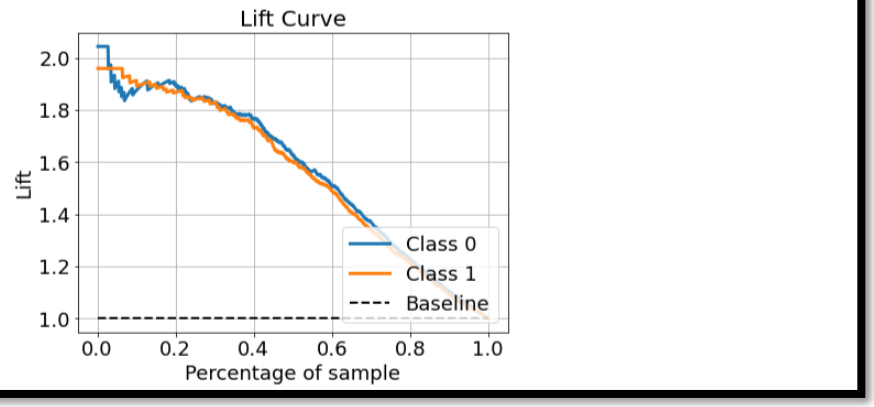


➢ Gradient Boosting Classification Model with Hyperparameter Tuning

```
In [128]: # Predict the classes on the test data, and return the probabilities for each class
          YY_proba = model1.predict_proba(X_test)
          skplt.metrics.plot_cumulative_gain(Y_test, YY_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
          plt.show()
```




```
In [129]: skplt.metrics.plot_lift_curve(Y_test, YY_proba, figsize=(7, 5), title_fontsize=20, text_fontsize=18)
          plt.show()
```



Cumulative gains and lift charts are visual aids for measuring model performance.
The Greater the area between the Lift / Gain and Baseline, the Better the model.
By analysing Gain and Lift Curve, Random Forest Classification Model, KNNC model and Gradient Boosting Classification model with Hyperparameter Tuning are the best models.

9.3. KS Statistics and ROC-AUC Score

In most binary classification problems, we use the KS-2samp test and ROC AUC score as measurements of how well the model separates the predictions of the two different classes.
The KS statistic for two samples is simply the highest distance between their two CDFs, so if we measure the distance between the positive and negative class distributions, we can have another metric to evaluate classifiers.
The ROC AUC score goes from 0.5 to 1.0, while KS statistics range from 0.0 to 1.0.

```
from scipy import stats
from scipy.stats import ks_2samp
from sklearn.metrics import roc_auc_score
```

```
def evaluate_ks_and_roc_auc(y_real, y_proba):
    # Unite both visions to be able to filter
    df = pd.DataFrame()
    df['real'] = y_real
    df['proba'] = y_proba[:, 1]

    # Recover each class
    class0 = df[df['real'] == 0]
    class1 = df[df['real'] == 1]

    ks = ks_2samp(class0['proba'], class1['proba'])
    roc_auc = roc_auc_score(df['real'] , df['proba'])

    print(f"KS: {ks.statistic:.4f} (p-value: {ks.pvalue:.3e})")
    print(f"ROC AUC: {roc_auc:.4f}")

    return ks.statistic, roc_auc
```

- Logistic Regression Model

```
In [131]: #Logistic Regression
          # Fit the model to the training data
          reg.fit(X_train,Y_train)
          # Predict the classes on the test data
          Y_pred=reg.predict(X_test)
          # Predict the classes on the test data, and return the probabilities for each class
          Y_proba = reg.predict_proba(X_test)
```

```
In [138]: print("Logistic Regression:")
          ks_LR, auc_LR = evaluate_ks_and_roc_auc(Y_test, Y_proba)

          Logistic Regression:
          KS: 0.5652 (p-value: 9.293e-65)
          ROC AUC: 0.8490
```

- Random Forest Classification Model

```
In [132]: #RandomForestClassifier
          # Fit the model to the training data
          clf.fit(X_train,Y_train)
          # Predict the classes on the test data
          Y1_pred=clf.predict(X_test)
          # Predict the classes on the test data, and return the probabilities for each class
          Y1_proba = clf.predict_proba(X_test)
```

```
In [139]: print("Random Forest classifier:")
          ks_RFC, auc_RFC = evaluate_ks_and_roc_auc(Y_test, Y1_proba)

          Random Forest classifier:
          KS: 0.5563 (p-value: 1.449e-62)
          ROC AUC: 0.8437
```

- KNN Classification Model

```
In [133]: #KNeighborsClassifier without hyperparameter tuning
          # Fit the model to the training data
          neigh.fit(X_train,Y_train)
          # Predict the classes on the test data
          YK_pred=neigh.predict(X_test)
          # Predict the classes on the test data, and return the probabilities for each class
          YK_proba = neigh.predict_proba(X_test)
```

```
In [140]: print("KNeighbors classifier:")
          ks_RFC, auc_RFC = evaluate_ks_and_roc_auc(Y_test, YK_proba)

          KNeighbors classifier:
          KS: 0.5741 (p-value: 5.306e-67)
          ROC AUC: 0.8500
```

- KNN Classification with Hyperparameter Tuning

```
In [134]: #KNeighborsClassifier with hyperparameter tuning
          # Fit the model to the training data
          modelkn.fit(X_train,Y_train)
          # Predict the classes on the test data
          Ykn_pred=modelkn.predict(X_test)
          # Predict the classes on the test data, and return the probabilities for each class
          Ykn_proba = modelkn.predict_proba(X_test)
```

```
In [141]: print("KNeighbors classifier with hyperparameter tuning:")
          ks_RFC, auc_RFC = evaluate_ks_and_roc_auc(Y_test, Ykn_proba)

          KNeighbors classifier with hyperparameter tuning:
          KS: 0.5741 (p-value: 5.306e-67)
          ROC AUC: 0.8500
```

- Gradient Boosting Classification Model

```
In [135]: #BoostingGradientClassifier without hyperparameter tuning
          # Fit the model to the training data
          model.fit(X_train,Y_train)
          # Predict the classes on the test data
          Y2_pred=model.predict(X_test)
          # Predict the classes on the test data, and return the probabilities for each class
          Y2_proba = model.predict_proba(X_test)
```

```
In [142]: print("Gradient Boosting classifier:")
          ks_GBC, auc_GBC = evaluate_ks_and_roc_auc(Y_test, Y2_proba)

          Gradient Boosting classifier:
          KS: 0.5564 (p-value: 1.354e-62)
          ROC AUC: 0.8530
```

- Gradient Boosting Classification Model with Hyperparameter Tuning

```
In [136]: #BoostingGradientClassifier with hyperparameter tuning
          # Fit the model to the training data
          model1.fit(X_train,Y_train)
          # Predict the classes on the test data
          YY_pred=model1.predict(X_test)
          # Predict the classes on the test data, and return the probabilities for each class
          YY_proba = model1.predict_proba(X_test)
```

```
In [143]: print("Gradient Boosting classifier with hyperparameter tuning:")
          ks_GBC, auc_GBC = evaluate_ks_and_roc_auc(Y_test, YY_proba)

          Gradient Boosting classifier with hyperparameter tuning:
          KS: 0.6243 (p-value: 1.022e-80)
          ROC AUC: 0.8888
```

10. Model Selection

Model	Score All Dataset	Score Train Dataset	Score Test Dataset	TN	FP	FN	TP
Logistic Regression	0.8022	0.785	0.774	335	83	110	326
Random Forest	0.8028	0.768	0.752	369	49	163	273
KNNC	0.837	0.8635	0.787	332	86	96	340
KNNC + Hyperparameter Tuning	0.837	0.863	0.787	332	86	96	340
Gradient Boosting	0.857	0.868	0.77	333	85	111	325
Gradient Boosting + Hyperparameter Tuning	0.943	0.962	0.803	343	75	93	343

The highest score for all dataset is of Gradient Boosting with hyperparameter tuning. But the difference between the score train dataset and test dataset is 16% which means its over fitting model. For the persistency of the drug the FN can cost a lot to the healthcare business, so it is needed to be low. KNNC and KNNC with hyperparameter tuning have low FN. KNNC is the best model. Let's check other metrics for evaluation.

Model	Precision	Recall	F1 score	Accuracy	KS Statics	AUC- ROC
Logistic Regression	0 - 0.75 1 - 0.80	0.80 0.75	0.78 0.77	0.77	0.565	0.85
Random Forest	0 – 0.69 1 – 0.85	0.88 0.63	0.78 0.72	0.75	0.556	0.844
KNNC	0 – 0.78 1 – 0.80	0.79 0.78	0.78 0.79	0.79	0.574	0.85
KNNC + Hyperparameter Tuning	0 – 0.78 1 – 0.80	0.79 0.78	0.78 0.79	0.79	0.574	0.85

The F1- Score, KS statistics and AUC-ROC metrics of KNNC model are the best. Therefore, **The best model for deployment is K- Nearest Neighbor Classification model.**

11. Save the Model

Last step is saving the model using pickle.

```
In [144]: # import pickle library
          import pickle # its used for seriealizing and de-seriealizing a python object Structure
          pickle.dump(neigh, open('model.pkl','wb'))       # open the file for writing
          model = pickle.load(open('model.pkl','rb'))    # dump an object to file object
```

12. Deployment of model into flask framework

A web application is developed that consists of a web page, after submitting the input in the form-based field to the web application, it will give the predicted persistency of the drug. Following is the directory structure of all files used for application.

12.1. App.py

The app.py file contains the source code including the ML code for prediction and will be execute by the Python interpreter to run the Flask web application.

```
import numpy as np
from flask import Flask, request, render_template
import pickle

#Create an app object using the Flask class.
app = Flask(__name__)

#Load the trained model. (Pickle file)
model = pickle.load(open('models/model.pkl', 'rb'))

#Define the route to be home.
#The decorator below links the relative route of the URL to the function it is decorating.
#Here, home function is with '/', our root directory.
#Running the app sends us to index.html.
#Note that render_template means it looks for the file in the templates folder.

#use the route() decorator to tell Flask what URL should trigger our function.
@app.route('/')
def home():
  return render_template('index.html')

#You can use the methods argument of the route() decorator to handle different HTTP methods.
#GET: A GET message is send, and the server returns data
#POST: Used to send HTML form data to the server.
#Add Post method to the decorator to allow for form submission.
#Redirect to /predict page with the output
@app.route('/predict',methods=['POST'])
def predict():

    int_features = [float(x) for x in request.form.values()] #Convert string inputs to float.
    features = [np.array(int_features)]  #Convert to the form [[a, b,c]] for input to the model
    prediction = model.predict(features)  # features Must be in the form [[a, b,c]]

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text="{}".format(output))


#When the Python interpreter reads a source file, it first defines a few special variables.
#For now, we care about the __name__ variable.
#If we execute our code in the main program, like in our case here, it assigns
# __main__ as the name (__name__).
#So if we want to run our code right here, we can check if __name__ == __main__
#if so, execute it here.
#If we import this file (module) to another file then __name__ == app (which is the name of this python file).

if __name__ == "__main__":
    app.run()
```

▪ Application will run as a single module; thus, a new Flask instance is initialized with the argument __name__ to let Flask know that it can find the HTML template folder (templates) in the same directory where it is located.

▪ Next, the route decorator is used (@app. route ('/')) to specify the URL that should trigger the execution of the home function. Home function simply rendered the index.html HTML file, which is in the templates folder.

▪ Predict function has the data set, it pre-processes the input, and make predictions, and then store the model. The input is entered by the user and uses the model to make a prediction for its label.

▪ The POST method is used to transport the form data to the server in the message body.

▪ The run function is used to only run the application on the server when this script is directly executed by the Python interpreter, which we ensured using the if statement with __name__ == '__main__'.

12.2. Index.html

The Index.html file will render a text form where a user enter the details of required fields. Index.html file will be rendered via the render_template ('index.html', prediction_text="{}".format(output)), which is inside the predict function of app.py script to display the output as per the input submitted by the user.

```
<!DOCTYPE html>

<html >
<head>
  <meta charset="UTF-8">

  <!-- Make it compatible to mobile devices -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>HEALTHCARE - PERSISTENCY OF THE DRUG</title>

</head>

<body style="background: #B5A642;">
 <div class="login">
    <h1 style="color:Black;">HEALTHCARE - PERSISTENCY OF THE DRUG</h1>

     <!-- Action is where the data is sent. In our case, predict page.
     If action is omitted, it assumed to be the current page -->
    <form action="{{ url_for('predict')}}"method="post">

        <label for="Region">Region:</label>
        <input type="text" name="Region" placeholder="Enter your Location" required="required" /><p>
        <h8 style="color:white;">Region:</h8><br>
        <h8 style="color:white;">0:Midwest, 1:Northeast, 2:Other/unknown, 3:South,
        4:West</h8><br></br>

        <label for="Ntm_Specialist_Flag">Ntm_Specialist_Flag:</label>
        <input type="text" name="Ntm_Specialist_Flag" placeholder="Select option " required="required" /><p>
        <h8 style="color:white;">Ntm_Specialist_Flag:</h8><br>
        <h8 style="color:white;">0:CARDIOLOGY, 1:CLINICAL NURSE SPECIALIST, 2:EMERGENCY MEDICINE,
         3:ENDOCRINOLOGY,4:GASTROENTEROLOGY, 5:GENERAL PRACTITIONER,</h8><br>
        <h8 style="color:white;"> 6:GERIATRIC MEDICINE, 7:HEMATOLOGY & ONCOLOGY, 8:HOSPICE AND PALLIATIVE MEDICINE,
        9:HOSPITAL MEDICINE, 10:NEPHROLOGY, </h8><br>
        <h8 style="color:white;">11:NEUROLOGY , 12:NUCLEAR MEDICINE, 13:OBSTETRICS & OBSTETRICS & GYNECOLOGY & OBSTETRICS & GYNECOLOGY,</h8><br>
        <h8 style="color:white;">14:OBSTETRICS AND GYNECOLOGY,15:OCCUPATIONAL MEDICINE, 16:ONCOLOGY,</h8><br>
        <h8 style="color:white;"> 17:OPHTHALMOLOGY, 18:ORTHOPEDIC SURGERY, 19:ORTHOPEDICS,
        20:OTOLARYNGOLOGY, </h8><br>
        <h8 style="color:white;">21:PAIN MEDICINE, 22:PATHOLOGY, 23:PEDIATRICS,
        24:PHYSICAL MEDICINE AND REHABILITATION, 25:PLASTIC SURGERY,</h8><br>
        <h8 style="color:white;"> 26:PODIATRY, 27:PSYCHIATRY AND NEUROLOGY,
        28:PULMONARY MEDICINE, 29:RADIOLOGY,30:RHEUMATOLOGY, </h8><br>
        <h8 style="color:white;">31:SURGERY AND SURGICAL SPECIALTIES, 32:TRANSPLANT SURGERY,
        33:UROLOGY,34:UNKNOWN, 35:VASCULAR SURGERY</h8><br></br>


        <label for="Ntm_Speciality_Bucket">Ntm_Speciality_Bucket:</label>
        <input type="text" name="Ntm_Speciality_Bucket" placeholder="Select option" required="required" /><p>
        <h8 style="color:white;">Ntm_Speciality_Bucket:</h8><br>
        <h8 style="color:white;">0:Endo/Onc/Uro, 1:OB/GYN/Others/PCP/Unknown, 2:Rheum</h8><br></br>
```

```
        <label for="Gluco_Record_During_Rx">Gluco_Record_During_Rx:</label>
        <input type="text" name="Gluco_Record_During_Rx" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Dexa_Freq_During_Rx">Dexa_Freq_During_Rx:</label>
        <input type="text" name="Dexa_Freq_During_Rx" placeholder="Enter the frequency" required="required" /><p><br>

        <label for="Dexa_During_Rx">Dexa_During_Rx:</label>
        <input type="text" name="Dexa_During_Rx" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Frag_Frac_During_Rx">Frag_Frac_During_Rx:</label>
        <input type="text" name="Frag_Frac_During_Rx" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Tscore_Bucket_Prior_Ntm">Tscore_Bucket_Prior_Ntm:</label>
        <input type="text" name="Tscore_Bucket_Prior_Ntm" placeholder="0:<=-2.5 / 1:>-2.5" required="required" /><p><br>

        <label for="Tscore_Bucket_During_Rx">Tscore_Bucket_During_Rx:</label>
        <input type="text" name="Tscore_Bucket_During_Rx" placeholder="Select option" required="required" /><p>
        <h8 style="color:white;">Tscore_Bucket_During_Rx:</h8><br>
        <h8 style="color:white;">0: <=-2.5, 1: >-2.5, 2:Unknown</h8><br></br>

        <label for="Change_T_Score">Change_T_Score:</label>
        <input type="text" name="Change_T_Score" placeholder="Select Option" required="required" /><p>
        <h8 style="color:white;">Change_T_Score:</h8><br>
        <h8 style="color:white;">0:Improved, 1:No Change, 2:Unknown, 3:Worsened </h8><br></br>

        <label for="Adherent_Flag">Adherent_Flag:</label>
        <input type="text" name="Adherent_Flag" placeholder="Select Option" required="required" /><p>
        <h8 style="color:white;">Adherent_Flag:</h8><br>
        <h8 style="color:white;">0:Adherent, 1:Non-Adherent</h8><br></br>

        <label for="Idn_Indicator">Idn_Indicator:</label>
        <input type="text" name="Idn_Indicator" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Injectable_Experience_During_Rx">Injectable_Experience_During_Rx:</label>
        <input type="text" name="Injectable_Experience_During_Rx" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Encounter_For_Screening_For_Malignant_Neoplasms">Comorb_Encounter_For_Screening_For_Malignant_Neoplasms:</label>
        <input type="text" name="Comorb_Encounter_For_Screening_For_Malignant_Neoplasms" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Encounter_For_Immunization">Comorb_Encounter_For_Immunization:</label>
        <input type="text" name="Comorb_Encounter_For_Immunization" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx">Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx:</label>
        <input type="text" name="Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Vitamin_D_Deficiency">Comorb_Vitamin_D_Deficiency:</label>
        <input type="text" name="Comorb_Vitamin_D_Deficiency" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified">Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified:</label>
        <input type="text" name="Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified" placeholder="0=No / 1=Yes" required="required" /><p><br>
```

```
        <label for="Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx">Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx:</label>
        <input type="text" name="Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Long_Term_Current_Drug_Therapy">Comorb_Long_Term_Current_Drug_Therapy:</label>
        <input type="text" name="Comorb_Long_Term_Current_Drug_Therapy" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Dorsalgia">Comorb_Dorsalgia:</label>
        <input type="text" name="Comorb_Dorsalgia" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Personal_History_Of_Other_Diseases_And_Conditions">Comorb_Personal_History_Of_Other_Diseases_And_Conditions:</label>
        <input type="text" name="Comorb_Personal_History_Of_Other_Diseases_And_Conditions" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Other_Disorders_Of_Bone_Density_And_Structure">Comorb_Other_Disorders_Of_Bone_Density_And_Structure:</label>
        <input type="text" name="Comorb_Other_Disorders_Of_Bone_Density_And_Structure" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias">Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias:</label>
        <input type="text" name="Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Osteoporosis_without_current_pathological_fracture">Comorb_Osteoporosis_without_current_pathological_fracture:</label>
        <input type="text" name="Comorb_Osteoporosis_without_current_pathological_fracture" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Personal_history_of_malignant_neoplasm">Comorb_Personal_history_of_malignant_neoplasm:</label>
        <input type="text" name="Comorb_Personal_history_of_malignant_neoplasm" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Comorb_Gastro_esophageal_reflux_disease">Comorb_Gastro_esophageal_reflux_disease:</label>
        <input type="text" name="Comorb_Gastro_esophageal_reflux_disease" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Concom_Cholesterol_And_Triglyceride_Regulating_Preparations">Concom_Cholesterol_And_Triglyceride_Regulating_Preparations:</label>
        <input type="text" name="Concom_Cholesterol_And_Triglyceride_Regulating_Preparations" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Concom_Narcotics">Concom_Narcotics:</label>
        <input type="text" name="Concom_Narcotics" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Concom_Systemic_Corticosteroids_Plain">Concom_Systemic_Corticosteroids_Plain:</label>
        <input type="text" name="Concom_Systemic_Corticosteroids_Plain" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Concom_Anti_Depressants_And_Mood_Stabilisers">Concom_Anti_Depressants_And_Mood_Stabilisers:</label>
        <input type="text" name="Concom_Anti_Depressants_And_Mood_Stabilisers" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Concom_Fluoroquinolones">Concom_Fluoroquinolones:</label>
        <input type="text" name="Concom_Fluoroquinolones" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Concom_Cephalosporins">Concom_Cephalosporins:</label>
        <input type="text" name="Concom_Cephalosporins" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Concom_Macrolides_And_Similar_Types">Concom_Macrolides_And_Similar_Types:</label>
        <input type="text" name="Concom_Macrolides_And_Similar_Types" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Concom_Broad_Spectrum_Penicillins">Concom_Broad_Spectrum_Penicillins:</label>
        <input type="text" name="Concom_Broad_Spectrum_Penicillins" placeholder="0=No / 1=Yes" required="required" /><p><br>
```

```
        <label for="Concom_Anaesthetics_General">Concom_Anaesthetics_General:</label>
        <input type="text" name="PConcom_Anaesthetics_General" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Concom_Viral_Vaccines">Concom_Viral_Vaccines:</label>
        <input type="text" name="Concom_Viral_Vaccines" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Type_1_Insulin_Dependent_Diabetes">Risk_Type_1_Insulin_Dependent_Diabetes:</label>
        <input type="text" name="Risk_Type_1_Insulin_Dependent_Diabetes" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Rheumatoid_Arthritis">Risk_Rheumatoid_Arthritis:</label>
        <input type="text" name="Risk_Rheumatoid_Arthritis" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Untreated_Chronic_Hyperthyroidism">Risk_Untreated_Chronic_Hyperthyroidism:</label>
        <input type="text" name="Risk_Untreated_Chronic_Hyperthyroidism" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Untreated_Chronic_Hypogonadism">Risk_Untreated_Chronic_Hypogonadism:</label>
        <input type="text" name="Risk_Untreated_Chronic_Hypogonadism" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Smoking_Tobacco">Risk_Smoking_Tobacco:</label>
        <input type="text" name="Risk_Smoking_Tobacco" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Chronic_Malnutrition_Or_Malabsorption">Risk_Chronic_Malnutrition_Or_Malabsorption:</label>
        <input type="text" name="Risk_Chronic_Malnutrition_Or_Malabsorption" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Chronic_Liver_Disease">Risk_Chronic_Liver_Disease:</label>
        <input type="text" name="Risk_Chronic_Liver_Disease" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Vitamin_D_Insufficiency">Risk_Vitamin_D_Insufficiency:</label>
        <input type="text" name="Risk_Vitamin_D_Insufficiency" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Poor_Health_Frailty">Risk_Poor_Health_Frailty:</label>
        <input type="text" name="Risk_Poor_Health_Frailty" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Excessive_Thinness">Risk_Excessive_Thinness:</label>
        <input type="text" name="Risk_Excessive_Thinness" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Hysterectomy_Oophorectomy">Risk_Hysterectomy_Oophorectomy:</label>
        <input type="text" name="Risk_Hysterectomy_Oophorectomy" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Estrogen_Deficiency">Risk_Estrogen_Deficiency:</label>
        <input type="text" name="Risk_Estrogen_Deficiency" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Immobilization">Risk_Immobilization:</label>
        <input type="text" name="Risk_Immobilization" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Risk_Recurring_Falls">Risk_Recurring_Falls:</label>
        <input type="text" name="Risk_Recurring_Falls" placeholder="0=No / 1=Yes" required="required" /><p><br>

        <label for="Count_Of_Risks">Count_Of_Risks:</label>
        <input type="text" name="Count_Of_Risks" placeholder="Enter the Counts" required="required" /><p>

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button></p>
```

```
            <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button></p>
        </form>

    </div>
        <div class="login">

        <h1 style="color:Red;">The Drug is: {{prediction_text}}</h1>
        </div>
        <hB style="color:white;">1 = Persistent </hB><br>
        <hB style="color:white;">0 = Non-Persistent </hB><br>
    </body>
    </html>
```

12.3. Development Server

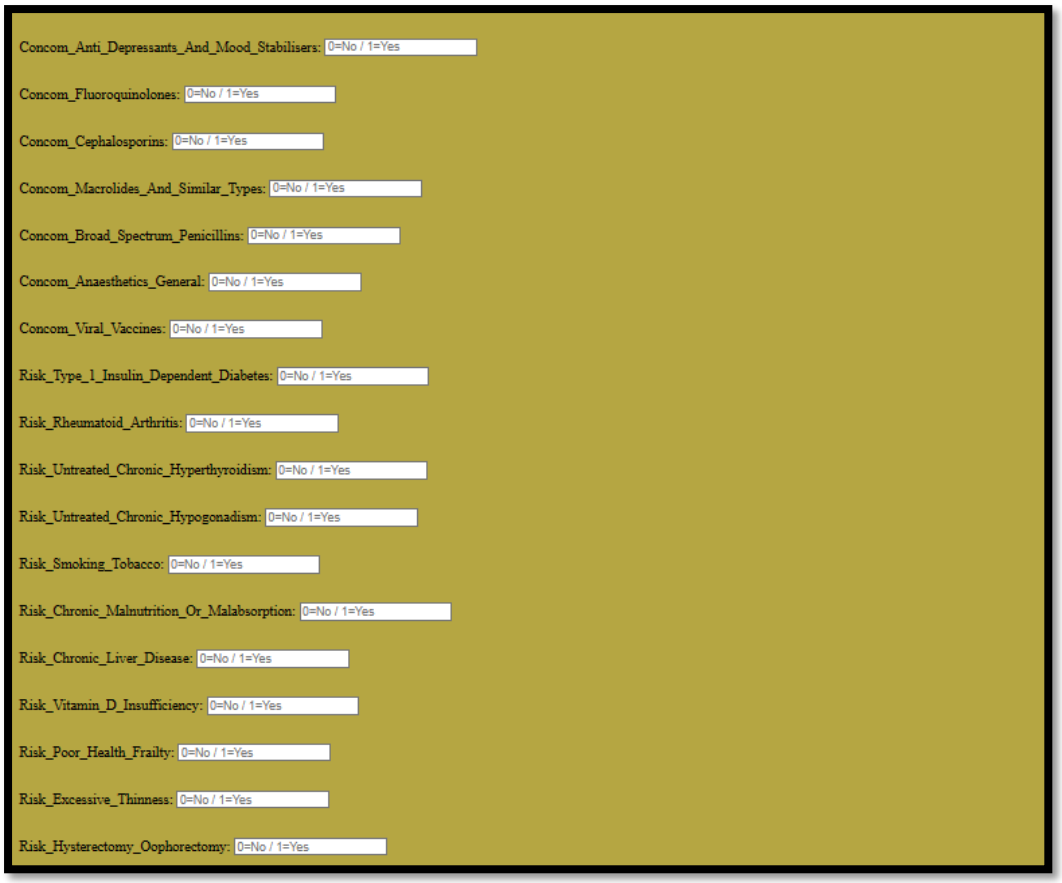
Following is the URL generate by 'app.py.'

```
In [2]: runfile('D:/Hira internship DataGlacier/final project/app.py', wdir='D:/
Hira internship DataGlacier/final project')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now open a web browser and navigate to http://127.0.0.1:5000/ following is output of Index.html.


127.0.0.1:5000
HEALTHCARE - PERSISTENCY OF THE DRUG
Region: Enter your Locaion
Region:
0:Midwest, 1:Northeast, 2:Other/unknown, 3:South, 4:West
Ntm_Specialist_Flag: Select option
Ntm_Specialist_Flag:
0:CARDIOLOGY, 1:CLINICAL NURSE SPECIALIST, 2:EMERGENCY MEDICINE, 3:ENDOCRINOLOGY,4:GASTROENTEROLOGY, 5:GENERAL PRACTITIONER,
6:GERIATRIC MEDICINE, 7:HEMATOLOGY & ONCOLOGY, 8:HOSPICE AND PALLIATIVE MEDICINE, 9:HOSPITAL MEDICINE, 10:NEPHROLOGY,
11:NEUROLOGY , 12:NUCLEAR MEDICINE, 13:OBSTETRICS & OBSTETRICS & GYNECOLOGY & OBSTETRICS & GYNECOLOGY,
14:OBSTETRICS AND GYNECOLOGY,15:OCCUPATIONAL MEDICINE, 16:ONCOLOGY,
17:OPHTHALMOLOGY, 18:ORTHOPEDIC SURGERY, 19:ORTHOPEDICS, 20:OTOLARYNGOLOGY,
21:PAIN MEDICINE, 22:PATHOLOGY, 23:PEDIATRICS, 24:PHYSICAL MEDICINE AND REHABILITATION, 25:PLASTIC SURGERY,
26:PODIATRY, 27:PSYCHIATRY AND NEUROLOGY, 28:PULMONARY MEDICINE, 29:RADIOLOGY,30:RHEUMATOLOGY,
31:SURGERY AND SURGICAL SPECIALTIES, 32:TRANSPLANT SURGERY, 33:UROLOGY,34:UNKNOWN, 35:VASCULAR SURGERY
Ntm_Speciality_Bucket: Select option
Ntm_Speciality_Bucket:
0:Endo/Onc/Uro, 1:OB/GYN/Others/PCP/Unknown, 2:Rheum
Gluco_Record_During_Rx: 0=No / 1=Yes
Dexa_Freq_During_Rx: Enter the frequency
Dexa_During_Rx: 0=No / 1=Yes
Frag_Frac_During_Rx: 0=No / 1=Yes
Tscore_Bucket_Prior_Ntm: 0:<=-2.5 / 1:>-2.5
Tscore_Bucket_During_Rx: Select option
Tscore_Bucket_During_Rx:
0: <=-2.5, 1: >-2.5, 2:Unknown
Change_T_Score: Select Option
Change_T_Score:
0:Improved, 1:No Change, 2:Unknown, 3:Worsened
Adherent_Flag: Select Option
Adherent_Flag:
0:Adherent, 1:Non-Adherent

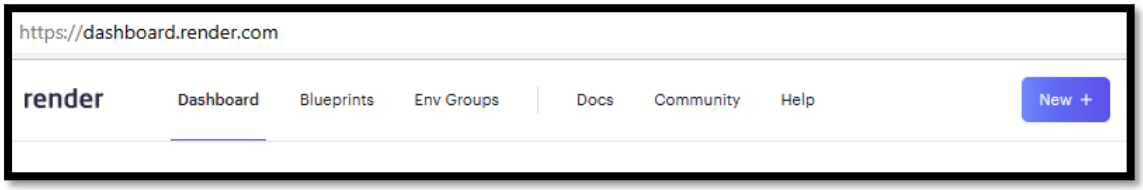

Idn_Indicator: 0=No / 1=Yes
Injectable_Experience_During_Rx: 0=No / 1=Yes
Comorb_Encounter_For_Screening_For_Malignant_Neoplasms: 0=No / 1=Yes
Comorb_Encounter_For_Immunization: 0=No / 1=Yes
Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx: 0=No / 1=Yes
Comorb_Vitamin_D_Deficiency: 0=No / 1=Yes
Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified: 0=No / 1=Yes
Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx: 0=No / 1=Yes
Comorb_Long_Term_Current_Drug_Therapy: 0=No / 1=Yes
Comorb_Dorsalgia: 0=No / 1=Yes
Comorb_Personal_History_Of_Other_Diseases_And_Conditions: 0=No / 1=Yes
Comorb_Other_Disorders_Of_Bone_Density_And_Structure: 0=No / 1=Yes
Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias: 0=No / 1=Yes
Comorb_Osteoporosis_without_current_pathological_fracture: 0=No / 1=Yes
Comorb_Personal_history_of_malignant_neoplasm: 0=No / 1=Yes
Comorb_Gastro_esophageal_reflux_disease: 0=No / 1=Yes
Concom_Cholesterol_And_Triglyceride_Regulating_Preparations: 0=No / 1=Yes
Concom_Narcotics: 0=No / 1=Yes
Concom_Systemic_Corticosteroids_Plain: 0=No / 1=Yes



Select categorical fields as per their respective number in the given code and click the Predict button. The predicted result will be displayed at the bottom of the web page.

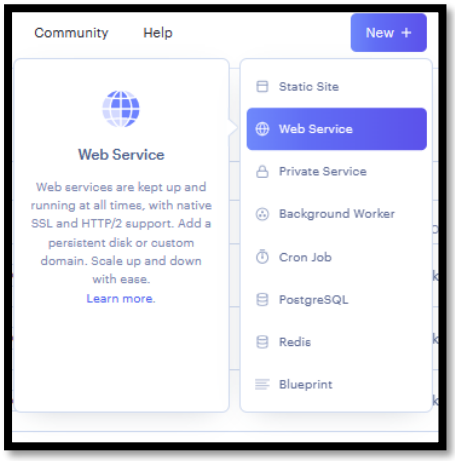
13. Model deployment on Render (Open-Source Cloud Deployment)

After the model has been trained and deployed locally, now it is ready for deploy on open-source cloud “Render”.



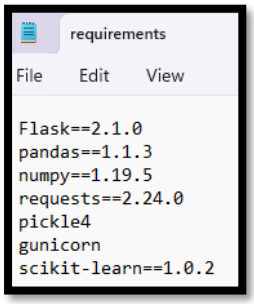
13.1. Web Service

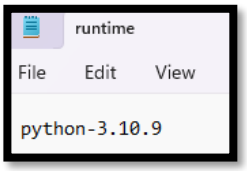
Click ‘New +’ then select ‘Web Service.’

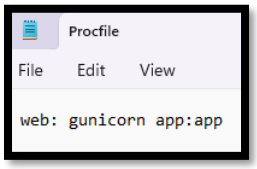


13.2. Connect to GitHub Repository

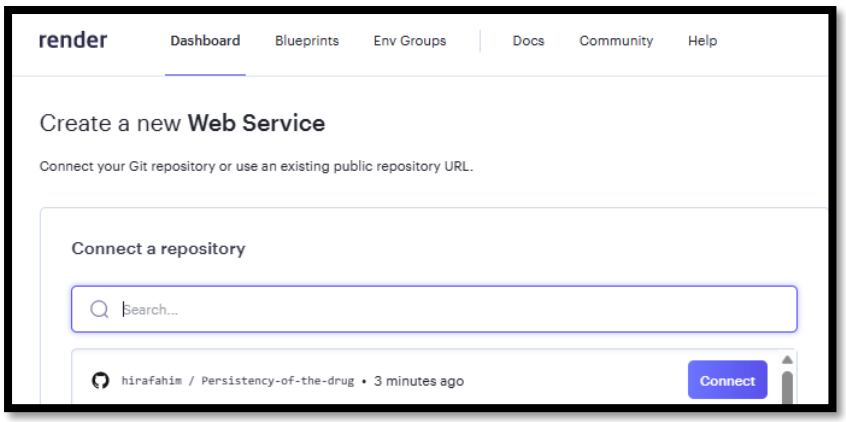
Before connecting to the GitHub repository, add required packages to the GitHub repository.

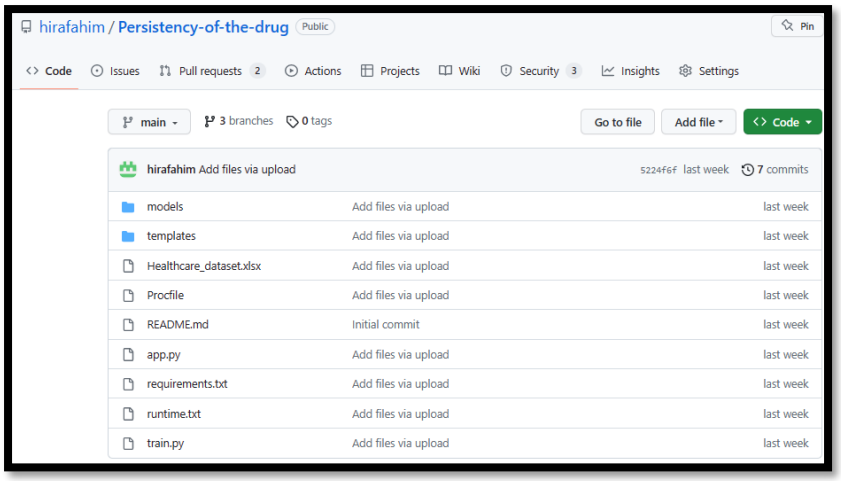




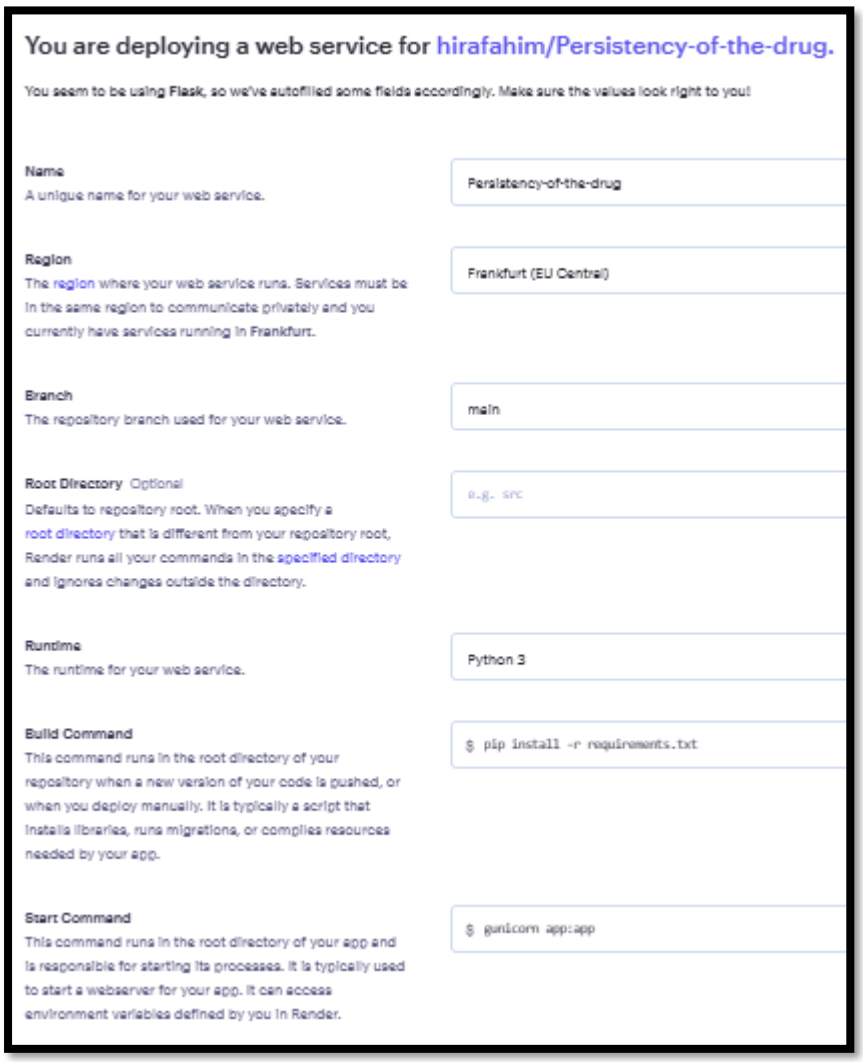


Connect to the GitHub repository.

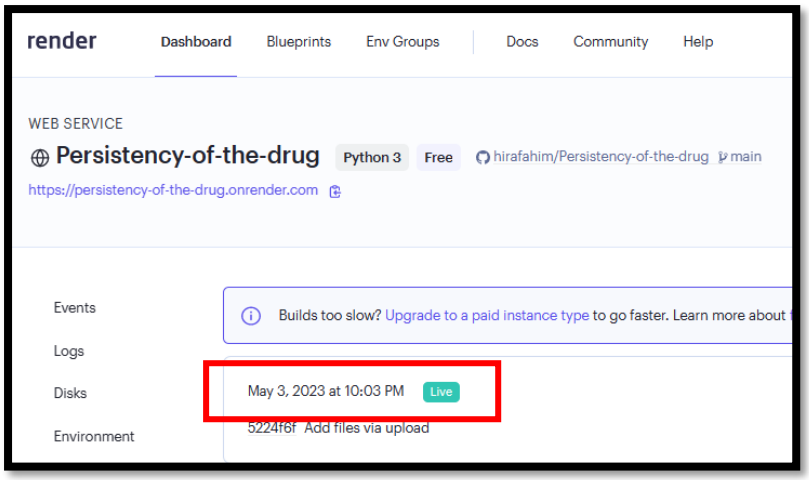


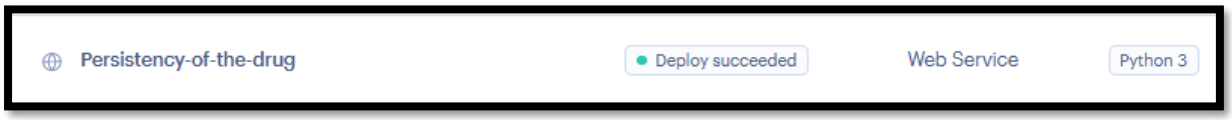


Fill in the required fields on Render dashboard and click the create to deploy the web app.



After 10 minutes, the web app is built and deployed successfully.

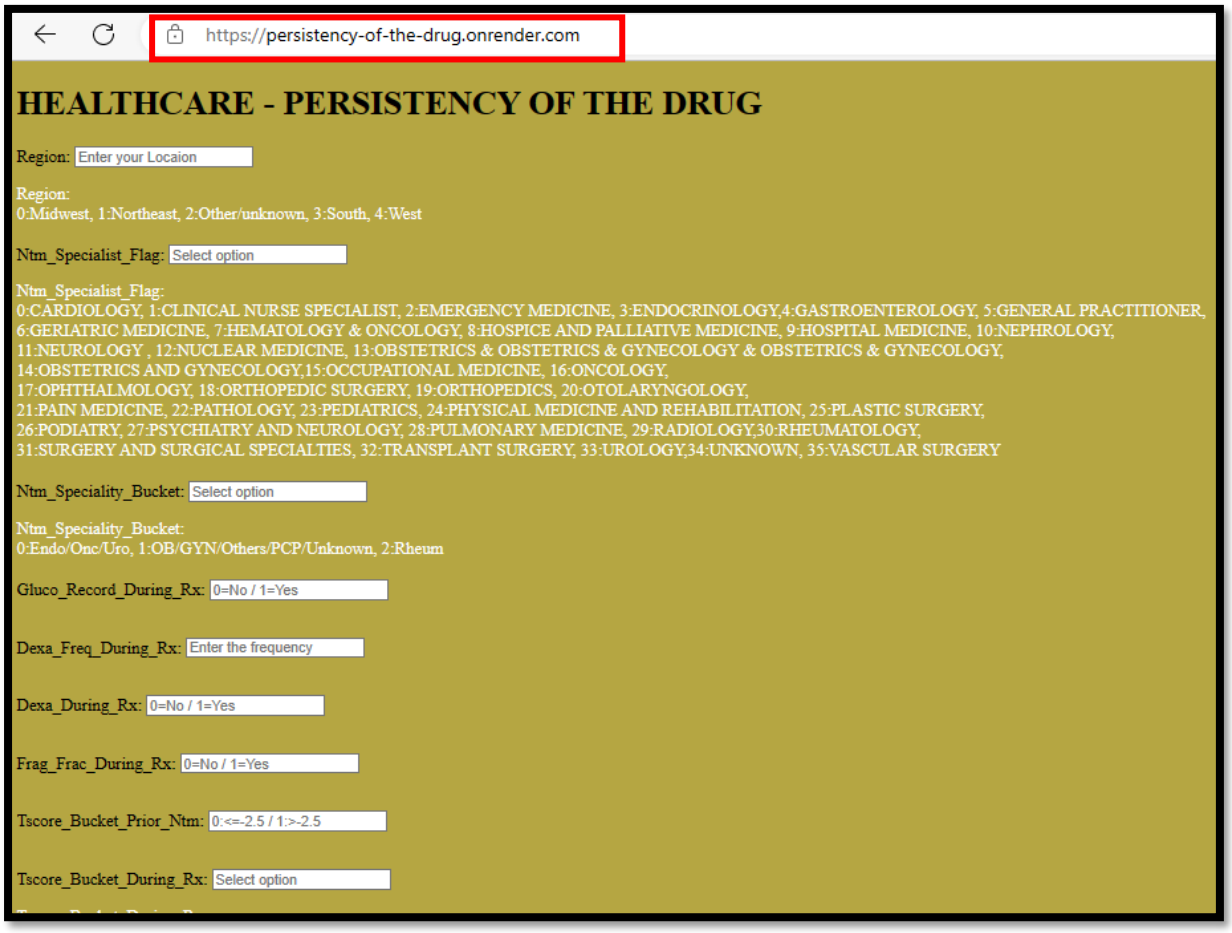


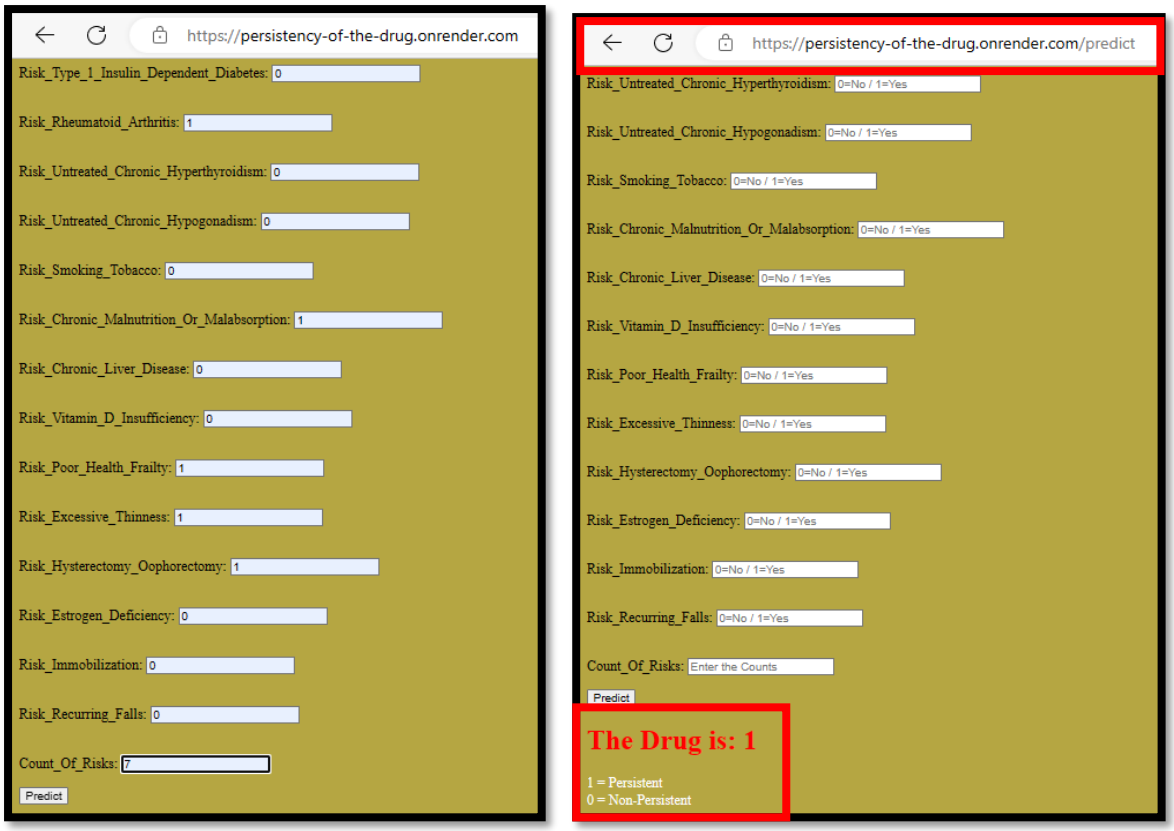


13.3. API- User Interface

This is the website link, click and open the application for persistency of the drug.

https://persistency-of-the-drug.onrender.com





14. Challenges

- Feature selection was a challenging task, which is done by Chi2 from sklearn.feature_selection library.
- Selection of best model was also tricky but after carefully considering all parameters and metrics of evaluation choose ‘K-Nearest Neighbor Classification model’ as the best model.