Data Science Intern at Data Glacier

Project: Healthcare - Persistency of a drug (Data Science)

Deliverables week 11- EDA PRESENTATION

NAMES	Taiwo Akingbesote	Kerr Tan	Farzana Chowdhury	Aya Ibrahim
UNIVERSITY	Montclair state University	New York University	Mount Holyoke College	University Of North Carolina
EMAIL	Akingbesotet1@montclair.edu	St4153@nyu.edu	Chowd23f@mtholyoke.edu	Ayariyadh9@gmail.com
COUNTRY	USA	USA	USA	USA
SPECIALIZATION	Data Science	Data Science	Data Science	Data Science
BATCH CODE	LISUM22	LISUM22	LISUM22	LISUM22
DATE	19 July 2023	19 July 2023	19 July 2023	19 July, 2023
ALL SUBMITTED TO DATA GLACIERS				

Table of Contents

1. Project Plan

WEEKS	DATE	PROGRESS
Week 7	July 19, 2023	Problem Statement and Data Processing
Week 8	July 26, 2023	Data preparation, Input, and output
Week 9	Aug 2, 2023	Analysis
Week 10	Aug 9, 2023	Model Approach and model Building
Week 11	Aug 16, 2023	EDA Presentation
Week 12	Aug 23, 2023	Flask development and Heroku
Week 13	Aug 30, 2023	Final project report and code

2. Problem Statement

The pharmaceutical industry faces numerous challenges in understanding the persistency of drugs as per physician prescriptions. Ensuring that patients adhere to prescribed medications is crucial for their health outcomes and overall treatment effectiveness. However, non-adherence to prescribed medications can lead to suboptimal results, increased healthcare costs, and potential complications. To address this critical issue, ABC pharma company has decided to take a data-driven approach and has approached us to automate the process of identifying factors impacting drug persistency.

Our task is to analyze a comprehensive dataset containing a diverse set of variables related to patient demographics, provider attributes, clinical factors, disease/treatment factors, comorbidities, concomitancy, and

adherence information. The primary objective is to gather insights and build a robust classification model that predicts whether a patient will exhibit persistency with the prescribed drug or not. The target variable, "Persistency_Flag," will serve as the ground truth for this classification task, where it is coded as 1 if the patient is persistent and 0 if the patient is nonpersistent between variables. Then build a model that classifies the dataset.

3. Data Collection

Healthcare_dataset.xlsx –

Total number of observations	27
Total number of files	2 – Feature Description and SDataset
Total number of features	27 & 3424
Base format of the file	xlsx
Size of the data	920kb

There was no missing values

4. Data Preparation

4.1 Data Manipulation - After collecting the data, we noticed that there is 'N' and 'Y' for most of the one-hot encoding categorical value, it needs to change to 0 and 1 in the following step to get an accurate overview of the data.

Before

```
data = pd.read_csv(sensitive_data_file)
data.head()
```

f_Osteoporosis	Risk_Low_Calcium_Intake	Risk_Vitamin_D_Insufficiency	Risk_Poor_Health_Frailty	Risk_Excessive_Thinness	Risk_
N	N	N	N	N	
N	N	N	N	N	
N	Y	N	N	N	
N	N	N	N	N	
N	N	N	N	N	

After

```
data = data.replace('N', 0)
data = data.replace('Y', 1)
data.head()
```

k_Family_History_Of_Osteoporosis	Risk_Low_Calcium_Intake	Risk_Vitamin_D_Insufficiency	Risk_Poor_Health_Frailty	Risk_Excessi
0	0	0	0	
0	0	0	0	
0	1	0	0	
0	0	0	0	
0	0	0	0	

4.2 Data Verification – After manipulating the data to getting an accurate view of our data, we verified the data types and check for any duplicate value(s)

```
[ ]  # Check if there are any duplicate values:

     duplicates = data.duplicated()

     if duplicates.any():
         print("Duplicates exist in the DataFrame.")
     else:
         print("No duplicates found in the DataFrame.")

     No duplicates found in the DataFrame.

[ ]  # Double check if there are any null (NA) values:

     Null_Values = data.isnull().sum()

     if Null_Values.any():
         print("Null values exist in the DataFrame.")
     else:
         print("No null values exist in the DataFrame.")

     No null values exist in the DataFrame.
```

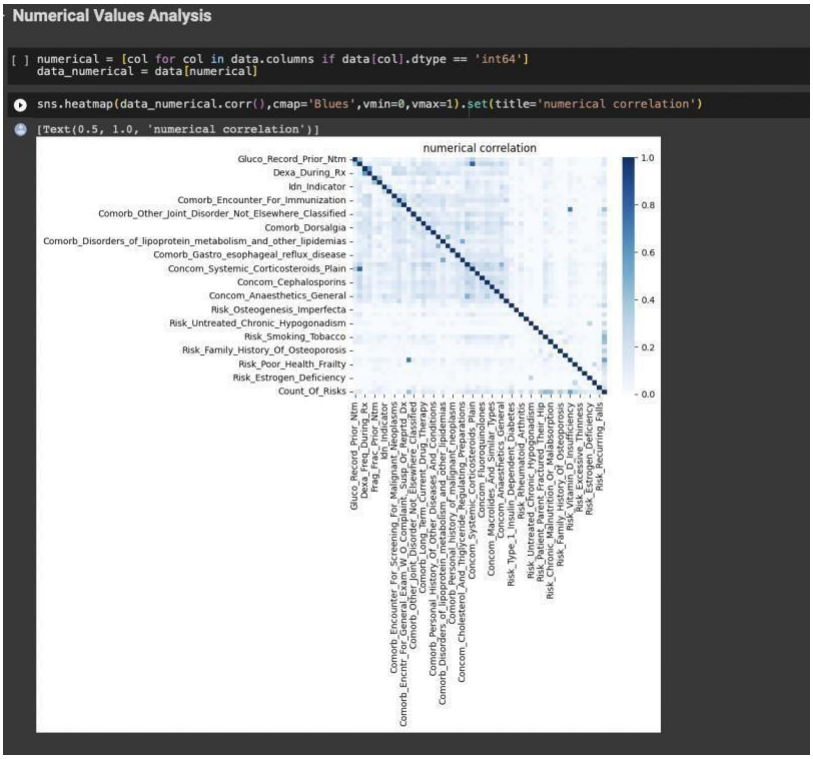
5. Analysis

To properly classify our data, we analyzed our dataset to have a deeper understanding of each category to give us a better approach to building our model. We analyzed the data into four different categories.

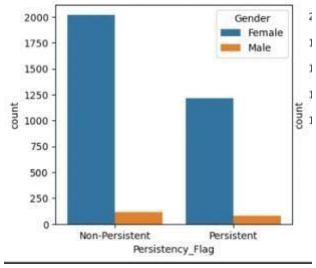
5.1 Numerical Values Analysis

Numerical Values Analysis

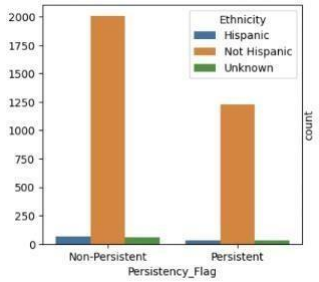
```
[ ] numerical = [col for col in data.columns if data[col].dtype == 'int64']
    data_numerical = data[numerical]

    sns.heatmap(data_numerical.corr(),cmap='Blues',vmin=0,vmax=1).set(title='numerical correlation')

    [Text(0.5, 1.0, 'numerical correlation')]
```



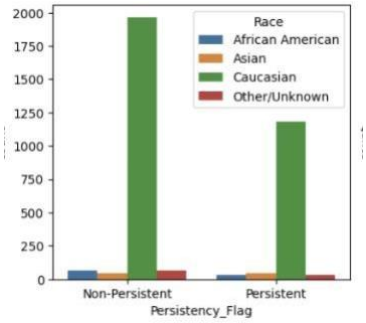
5.2 Patient Demographic Analysis – We grouped the patients by their gender, Ethnicity, Race, Region, and Age then plotted a graph for better visualization.



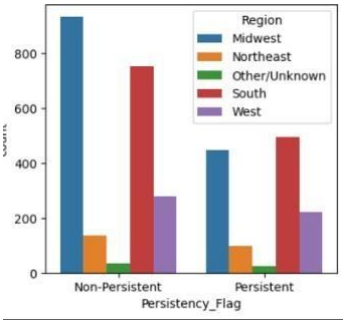
Gender



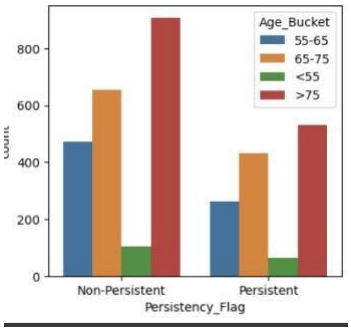
Ethnicity



Race



Region



Age

Findings

Based on the graphs above:

1) There are higher persistency and non-persistency counts in Females than in Males with non-persistency being higher.

2) There are higher persistency and non-persistency counts in Caucasians among all other races with no persistency being higher.

3) There are higher persistency and non-persistency counts in non-Hispanic people among all other ethnicities with non-persistency being higher.

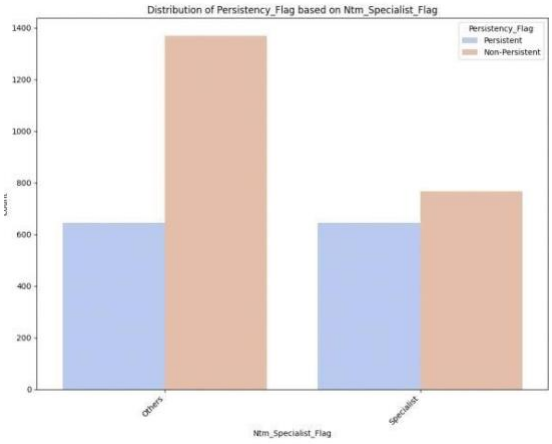
4) The highest persistency counts in order among regions is in the South, Midwest, and West regions. And the highest non-persistency counts in order is in the Midwest, South, and West regions.

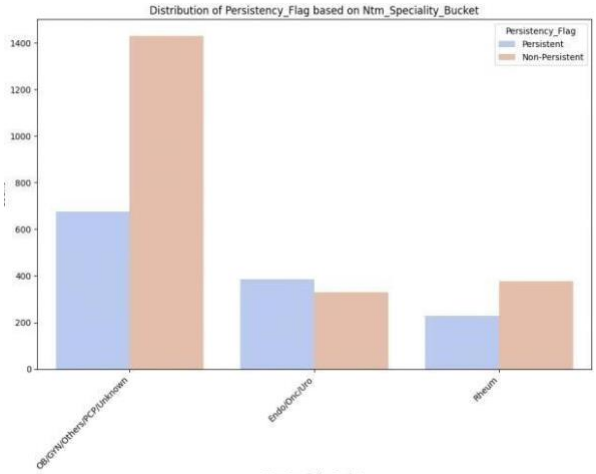
5) The highest persistency and non-persistency count in order are among patients of the following age groups: >75, 65-75, and 55-65.

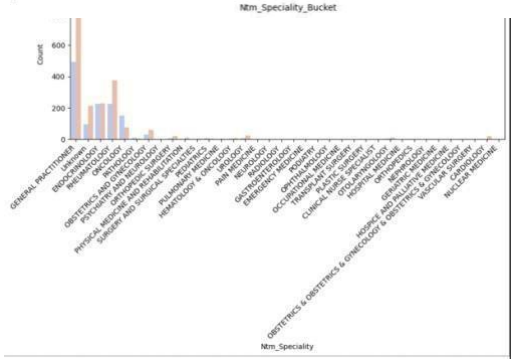
5.3 Physician/Provider Analysis - NTM Rx is the medication prescribed by the physician.

- Ntm_Speciality: Physician Specialty
- Ntm_Specialist_Flag: If physician is a specialist or not.
- Ntm_Speciality_Bucket: has 3 separate groups:
    - OB/GYN/Others/PCP/Unknown = Obstetrics and Gynecology, other specialties, Primary Care Physicians, and those of unknown specialty.
    - Endo/Onc/Uro: Endocrinology, Oncology, and Urology
    - Rheu: Rheumatology

We plotted three graphs to check if the physician's categorical features impact the persistency flag based on the categories above.







FINDINGS

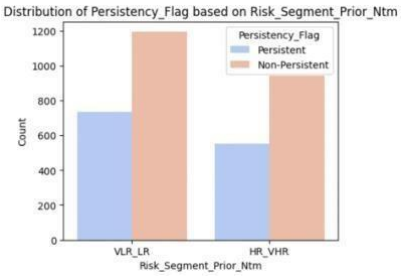
According to the graphs above:

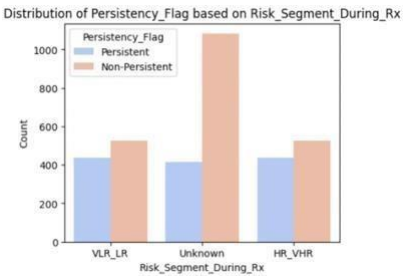
1) The highest non-persistency and persistency counts among patients occurred with those whose providers are general practitioners and rheumatologists with endocrinologists and unknown specialties coming next.
2) The highest non-persistency and persistency counts also occurred among patients whose providers' flag was categorized as non-specialists.
3) The highest non-persistency and persistency counts also occurred among patients whose providers' bucket was categorized as OB/GYN/Others/PCP/Unknown.

Based on this information, it's hard to detect what specialty led to the most non-persistency. However, generally, those who were general practitioners or non-specialists had higher non-persistency patient counts.

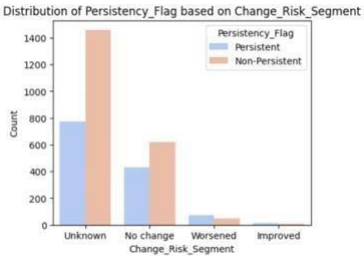
5.4 Risk Factors and Change, Adherence to Therapy, & T-score Change Analysis

With this analysis, we looked for other factors that may influence target variable and plotted their graphs to visualize them. They include. o Risk_Segment_Prior_Ntm & Risk_Segment_During_Rx: The risk segment of patients before they started their treatment (prior to receiving the NTM medication) with VLR_LR and HR_VHR representing Very Low Risk/Low Risk and Very High Risk/High Risk respectively.

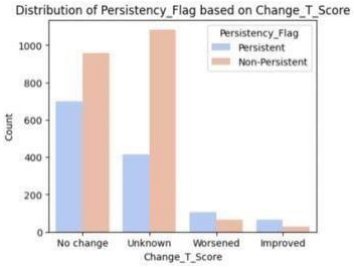




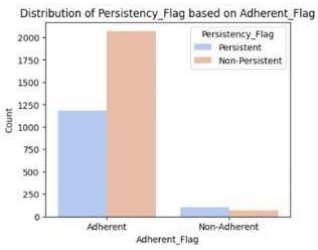
o Change_Risk_Segment: If there was any change in Risk Segment.



o Change_T_Score: The Tscore is a measurement used to assess bone density in the context of osteoporosis. This value indicates the change in the patient's bone density relative to that of a healthy adult.



o Adherent Flag: Adherence status of patients to their prescribed therapies and whether they followed the prescribed medication.



FINDINGS

According to graphs above:

1) There are higher non-persistency counts among patients who have low risk factors prior to taking their medication.

2) There are higher non-persistency counts among patients who have low risk and unknown factors during taking their medication.

3) Both the change in risk segment and the change in T-score are mostly either unknown or had no change with the count being higher for patients who were non-persistent.

4) Although many patients were adherent to their medication, there was still higher non-persistency among them.

6. MODEL APPROACH AND MODEL BUILDING

After our analysis, we concluded that using the demographic analysis, physician analysis and Risk Factors and Change, Adherence to Therapy, & T-score Change Analysis.

6.1 FEATURE ENGINEERING

This part includes formatting important data that proved to be very sensitive in determine the persistency.

- Feature Engineering using demographics

```
# Feature Engineering using demographics
data['IsFemale'] = data['Gender'].apply(lambda gender: 1 if gender == 'Female' else 0)
data['IsCaucasian'] = data['Race'].apply(lambda race: 1 if race == 'Caucasian' else 0)
data['IsNonHispanic'] = data['Ethnicity'].apply(lambda ethnicity: 1 if ethnicity == 'Not Hispanic' else 0)
data['IsAgeGroup1'] = data['Age_Bucket'].apply(lambda age_bucket: 1 if age_bucket == '>75' else 0)
data['IsAgeGroup2'] = data['Age_Bucket'].apply(lambda age_bucket: 1 if age_bucket == '65-75' else 0)
data['IsAgeGroup3'] = data['Age_Bucket'].apply(lambda age_bucket: 1 if age_bucket == '55-65' else 0)
```

- Feature Engineering using physician analysis

```
# Feature Engineering using physician analysis
data['IsGeneralPractitioner'] = data['Ntm_Speciality'].apply(lambda specialty: 1 if specialty == 'GENERAL PRACTITI
data['IsNonSpecialist'] = data['Ntm_Specialist_Flag'].apply(lambda flag: 1 if flag == 'Others' else 0)
```

- Feature Engineering using risk factors and adherence analysis

```
# Feature Engineering using risk factors and adherence analysis
data['IsLowRiskPrior'] = data['Risk_Segment_Prior_Ntm'].apply(lambda risk: 1 if risk == 'VLR_LR' else 0)
data['IsLowRiskDuring'] = data['Risk_Segment_During_Rx'].apply(lambda risk: 1 if risk == 'Unknown' else 0)
data['IsChangeRiskUnknown'] = data['Change_Risk_Segment'].apply(lambda change: 1 if change == 'Unknown' else 0)
data['IsChangeTScoreUnknown'] = data['Change_T_Score'].apply(lambda change: 1 if change == 'Unknown' or 'No Change
data['IsAdherent'] = data['Adherent_Flag'].apply(lambda flag: 1 if flag == 'Adherent' else 0)
```

6.2 Encoding Categories

After feature engineering, we encode the regio, ntm speciality and persistency flag and created a features list that takes users input

```
# Apply LabelEncoder to categorical columns
label_encoder = LabelEncoder()

data['Region'] = label_encoder.fit_transform(data['Region'])
data['Ntm_Speciality'] = label_encoder.fit_transform(data['Ntm_Speciality'])
data['Persistency_Flag'] = label_encoder.fit_transform(data['Persistency_Flag'])

features = ['IsFemale', 'IsCaucasian', 'IsNonHispanic', 'IsAgeGroup1', 'IsAgeGroup2',
            'IsAgeGroup3', 'IsGeneralPractitioner','IsNonSpecialist', 'IsOBGYNorPCP',
            'IsLowRiskPrior', 'IsLowRiskDuring', 'IsChangeRiskUnknown', 'IsChangeTScoreUnknown', 'IsAdherent']
```

6.3 Splitting the dataset.

- Spliting the dataset into train and test

```
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

6.4 Creating, training and testing the model.

```
# Create and train the model (Random Forest Classifier)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
print("X_train", X_train.shape)
print("y_train", y_train.shape)
print("X_test", X_test.shape)
print("y_test", y_test.shape)
```

```
X_train (2739, 14)
y_train (2739,)
X_test (685, 14)
y_test (685,)
```

6.5 Building the model and Evaluating the Model using the confusion matrix

```
# Make predictions
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.635036496350365
```

Evaluating the Model using the confusion matrix

```
from sklearn.metrics import confusion_matrix
# Evaluate the model
class_names = label_encoder.classes_


print(confusion_matrix(y_test, y_pred))
```

```
[[366  65]
 [183  71]]
```

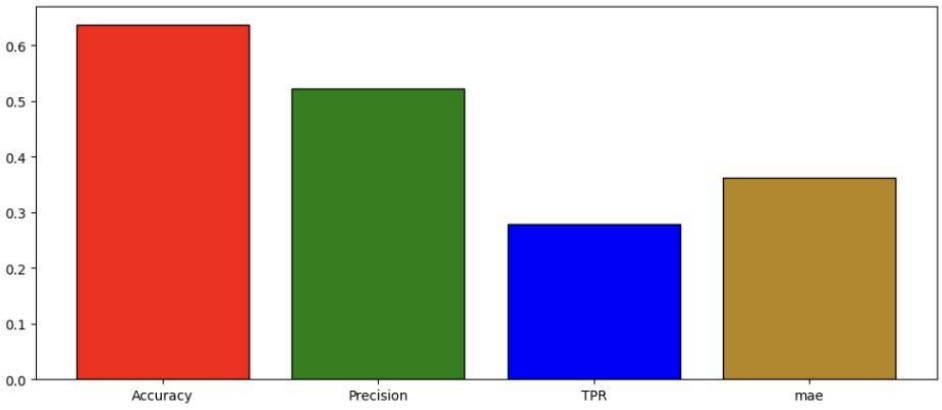
6.6 Calculating and visualizing mode accuracy

Our model was 63.7% accurate.

```
from sklearn.metrics import accuracy_score, recall_score

# Accuracy Score
Accuracy = accuracy_score(y_test, y_pred)
print('Accuracy Score:', Accuracy)

# Precision Score
Precision = precision_score(y_test, y_pred)
print('Precision Score:', Precision)

# True positive Rate (TPR) or Sensitivity or Recall
TPR = recall_score(y_test, y_pred)
print('True positive Rate:', TPR)


# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error:", mae)
```

```
Accuracy Score: 0.637956204379562
Precision Score: 0.5220588235294118
True positive Rate: 0.2795275590551181
Mean Absolute Error: 0.362043795620438
```

```
plt.figure(figsize = (12, 5))

result = [Accuracy, Precision, TPR, mae]
label = ["Accuracy", "Precision", 'TPR', 'mae']
colors=[ 'red', 'green', 'blue', 'darkgoldenrod', 'orange']

plt.bar(label, result, color = colors, edgecolor='black')
plt.show()
```



6.7 Saving the model using pickle.

```
import pickle
from sklearn.ensemble import RandomForestClassifier


model_filename = 'persistency_model.pkl'
with open(model_filename, 'wb') as model_file:
    pickle.dump(model, model_file)


with open(model_filename, 'rb') as model_file:
    loaded_model = pickle.load(model_file)
```

6.8 Creating a dictionary to store FEATURE values

```
# Define the list of features

features = ['IsFemale', 'IsCaucasian', 'IsNonHispanic', 'IsAgeGroup1', 'IsAgeGroup2',
            'IsAgeGroup3', 'IsGeneralPractitioner','IsNonSpecialist', 'IsOBGYNorPCP',
            'IsLowRiskPrior', 'IsLowRiskDuring', 'IsChangeRiskUnknown', 'IsChangeTScor

# Create an empty dictionary to store feature values
data_dict = {}

# Prompt the user to input feature values
print("Please provide the following feature values:")

for feature in features:
    if feature == 'IsFemale':
        value = int(input("If patient is female, enter 1. If male, enter 0: "))
    elif feature == 'IsCaucasian':
        value = int(input("If patient caucasian, enter 1. If not, enter 0: "))
    elif feature == 'IsNonHispanic':
        value = int(input("If patient is Hispanic, enter 1. If not, enter 0: "))
    elif feature == 'IsAgeGroup1':
        value = int(input("If patient age is >75, enter 1. If not, enter 0: "))
    elif feature == 'IsAgeGroup2':
        value = int(input("If patient age is between 65-75, enter 1. If not, enter 0:
    elif feature == 'IsAgeGroup3':
        value = int(input("If patient is age is between 55-65, enter 1. If not, enter
    elif feature == 'IsGeneralPractitioner':
        value = int(input("If patient's speciality is general practitioner, enter 1. I
    elif feature == 'IsNonSpecialist':
        value = int(input("If patient's physician is a Non specialist, enter 1. If not
    elif feature == 'IsOBGYNorPCP':
        value = int(input("If patient's provider is an Obstetrics or Gynecology, enter
    elif feature == 'IsLowRiskPrior':
        value = int(input("If patient's Risk_Segment_Prior_Ntm is VLR_LR, enter 1. If
    elif feature == 'IsLowRiskDuring':
        value = int(input("If patient's Risk_Segment_During_Rx is Unknown, enter 1. If
    elif feature == 'IsChangeRiskUnknown':
        value = int(input("If patient's Change_Risk_Segment is Unknown, enter 1. If nc
    elif feature == 'IsChangeTScoreUnknown':
        value = int(input("If patient's Change_T_Score is Unknown or No change, enter
    elif feature == 'IsAdherent':
        value = int(input("If patient's Adherent_Flag is Adherent, enter 1. If not, er
    elif feature.startswith('Is'):
        # For other binary features, prompt for a binary value (0 or 1)
        value = int(input(f"If Patient is {feature} enter 1 else enter 0): "))
    else:
        # For other features, prompt for a value
        value = input(f"Enter value for {feature}: ")
    data_dict[feature] = value

# Create a DataFrame from the user input
user_input_data = pd.DataFrame(data_dict, index=[0])

# Display the user input data
print("\nUser Input Data:")
print(user_input_data)
```

6.9 Testing the model

```
# Load the trained model from file
with open(model_filename, 'wb') as model_file:
    pickle.dump(model, model_file)


# Make predictions using the loaded model
prediction = loaded_model.predict(user_input_data[features])

# Display the prediction
if prediction[0] == 1:
    print("Prediction: Persistent")
else:
    print("Prediction: Non-Persistent")
```

```
Prediction: Non-Persistent
```