Group Project Deliverable Week 13 (Final Report)

Project Name: Healthcare - Persistency of a drug
Report date: 26.09.2023
Internship Batch: LISUM24
Group: HealthData Collective
Data storage location: https://github.com/venusflytrapfairy/HealthDataCollective-Group

Contents of Final Report

- **Team member's details**

Group: HealthData Collective
Group members:

Name	Email	Country	College/Company	Specialization
Selena	selena.elensto@gmail.com	Macedonia	-/-	Data Science
Anusha Asim	anushaasim21@yahoo.com	UAE	Regent Middle East	Data Science
Sukurat	salamsukurat@gmail.com	United Kingdom	Teesside University	Data Science
Khushboo	Khushboomasih193@gmail.com	United Kingdom	University of Aberdeen	Data Science

- **Problem description**

The problem to be solved is related to pharmaceutical companies' need to understand the persistence of drug usage among patients based on physician prescriptions. Specifically, the challenge is to automate the process of identifying whether a patient is persistent in following their prescribed therapy regimen. The primary problem is to build a classification model using machine learning to determine the persistence of patients based on various features and factors.

- **EDA performed on the data**

Our exploratory data analysis covered several key steps including data loading, data profiling, data cleaning, and feature engineering.

Here's a summary of the steps and tasks performed:

- *Data Loading:* We loaded the dataset from an Excel file using the Pandas library.

- *Data Profiling:* The explanations for each feature in the dataset was provided to help understand the meaning and relevance of each variable.

- *Data Cleaning:* We identified and handled the presence of "Unknown" values in the dataset, which can be crucial for data preprocessing.

- *Deduplication:* The potential duplicate records in the dataset were identified and handled.

- *Data Visualization:* We generated a profile report using the Pandas Profiling library to gain insights into the dataset's statistics and distributions.

- *Label Encoding:* The non-numerical features were encoded, including categorical and boolean features, in preparation for modeling.

- *Model Development:* We performed basic linear regression modeling and generated a summary using the statsmodels library.

- *Train-Test Split:* We split the dataset into training and testing sets.

- *Dummy Classifier:* A Dummy Classifier to create a baseline model and evaluate its performance on both the original and encoded datasets.

- *Oversampling:* We applied oversampling techniques such as SMOTE, ADASYN, SMOTETomek, and SMOTEENN to address the issue of imbalanced target classes.

- *Results Summary:* The results of each oversampling technique were collected and summarized, including the shape of the oversampled data and the accuracy score of a Dummy Classifier.

- *Comparison with Original Dataset:* We've included a row in the summary to compare results with the original dataset without oversampling.

This EDA serves as the foundation for subsequent modeling and decision-making in the context of medication persistence.

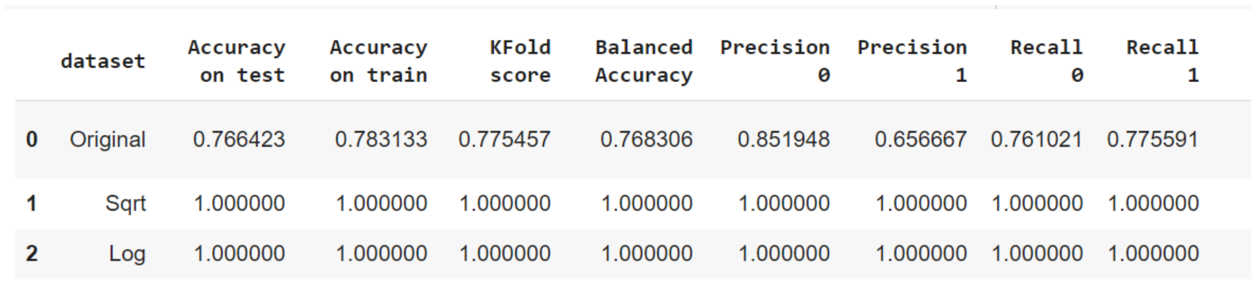
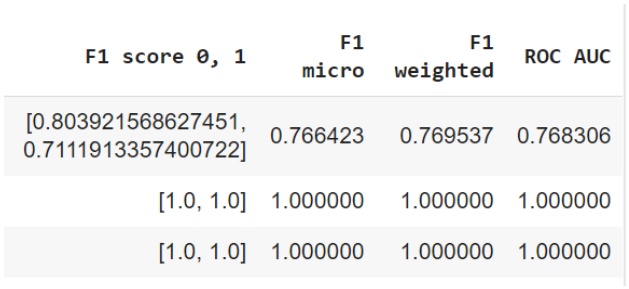
<u>Implementation of Models</u>

In this section, we delve into the application of machine learning models to address the challenge of medication persistence in healthcare. We explored a variety of models to discern their suitability for our dataset and problem statement:

- *Gaussian Naive Bayes Classifier:*

We chose to include the Gaussian Naive Bayes classifier in our modeling suite due to its specific capabilities and suitability for our dataset. Gaussian Naive Bayes is a probabilistic classification algorithm that is particularly effective when dealing with datasets that contain continuous (numerical) features. Our dataset, which includes variables like patient age, T-score, and DEXA scan frequency, consists of several numerical attributes, making Gaussian Naive Bayes an appropriate choice.

The key advantage of Gaussian Naive Bayes is its simplicity and efficiency. It is based on the assumption that the features are normally distributed, and it uses Gaussian probability distributions to model the data. In our context, this makes it well-suited for variables like age and T-score, which often follow a normal distribution pattern

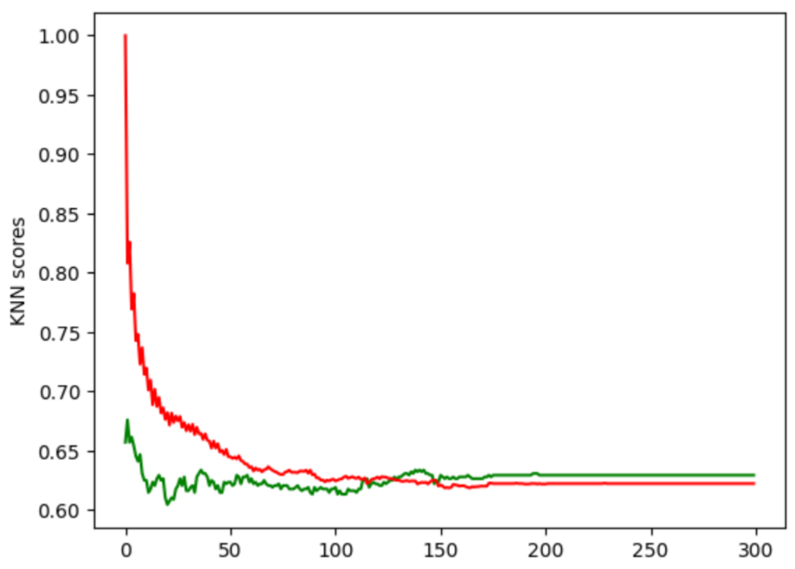



	dataset	Accuracy on test	Accuracy on train	KFold score	Balanced Accuracy	Precision 0	Precision 1	Recall 0	Recall 1	F1 score 0, 1	F1 micro	F1 weighted	ROC AUC
0	Original	0.766423	0.783133	0.775457	0.768306	0.851948	0.656667	0.761021	0.775591	[0.803921568627451, 0.7111913357400722]	0.766423	0.769537	0.768306
1	Sqrt	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	[1.0, 1.0]	1.000000	1.000000	1.000000
2	Log	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	[1.0, 1.0]	1.000000	1.000000	1.000000

- *K-Nearest Neighbors (KNN) Classifier:*

  KNN is a non-parametric, instance-based learning algorithm that is particularly useful when dealing with datasets that may exhibit complex, non-linear relationships between features. In our dataset, we have a mix of numerical and categorical attributes, making KNN a valuable choice for exploring interactions and patterns.

```
Max KNN scores is 0.6759124087591241 for k_neighbors: 2
```



	dataset	Accuracy on test	Accuracy on train	KFold score	Balanced Accuracy	Precision 0	Precision 1	Recall 0	Recall 1
0	Original	0.675912	0.807959	0.681640	0.590478	0.678632	0.660000	0.921114	0.259843
1	Sqrt	0.700730	0.828770	0.706452	0.622326	0.697552	0.716814	0.925754	0.318898
2	Log	0.699270	0.832786	0.710107	0.623591	0.699115	0.700000	0.916473	0.330709

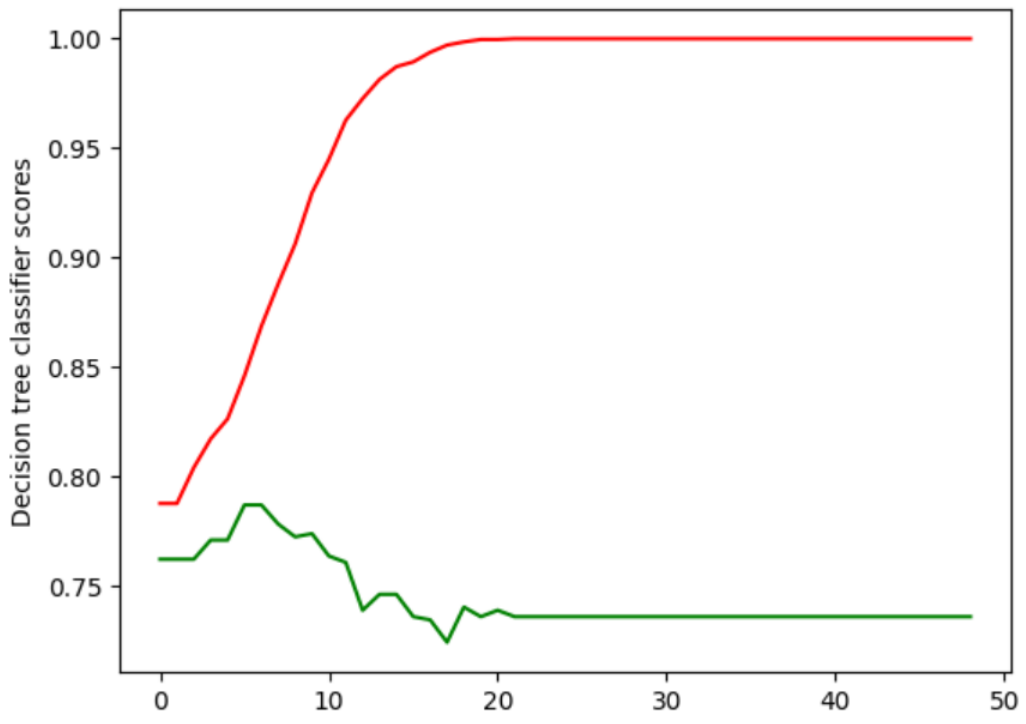
F1 score 0, 1	F1 micro	F1 weighted	ROC AUC
[0.7814960629921262, 0.37288135593220334]	0.675912	0.629981	0.590478
[0.7956131605184448, 0.44141689373297005]	0.700730	0.664276	0.622326
[0.7931726907630522, 0.44919786096256686]	0.699270	0.665626	0.623591

- *Decision Tree Classifier:*

The Decision Tree Classifier was included in our modeling approach due to its capacity to unravel the intricacies of medication persistence as related to physician prescriptions.

By creating a hierarchical structure of decisions based on patient attributes, this model can discern the factors that influence whether a patient follows their prescribed therapy.

```
Max Decision tree classifier scores is 0.7868613138686131 for tree depth = 6
```



	dataset	Accuracy on test	Accuracy on train	KFold score	Balanced Accuracy	Precision 0	Precision 1	Recall 0	Recall 1
0	Original	0.786861	0.845929	0.797001	0.748977	0.792608	0.772727	0.895592	0.602362
1	Sqrt	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
2	Log	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

F1 score 0, 1	F1 micro	F1 weighted	ROC AUC
[0.840958605664488, 0.6769911504424779]	0.786861	0.780159	0.748977
[1.0, 1.0]	1.000000	1.000000	1.000000
[1.0, 1.0]	1.000000	1.000000	1.000000

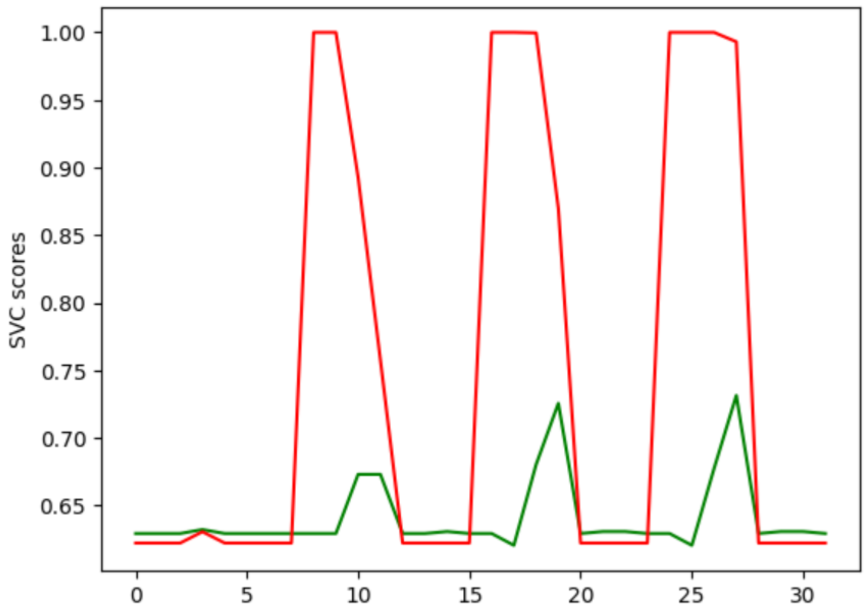
- *Support Vector Classifier (SVC):*

The Support Vector Classifier (SVC) was chosen as one of the models due to its suitability for binary classification tasks, which aligns with our primary objective of predicting medication persistence among patients based on various factors and attributes.

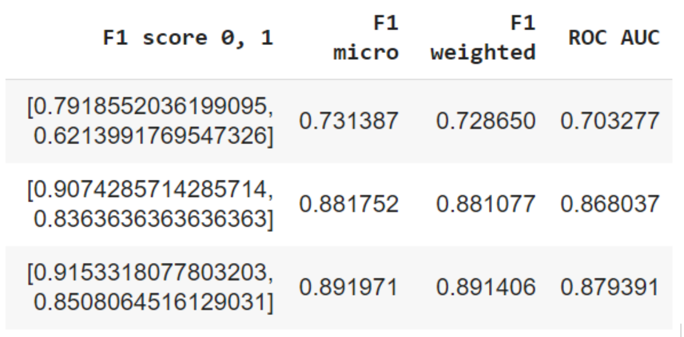
In the context of our healthcare problem, our primary concern is to classify patients into two categories: those who are persistent in following their prescribed therapy and those who are non-persistent. The SVC is particularly well-suited for this task, as it aims to establish an optimal decision boundary that effectively separates these two classes.

SVC can accommodate both linear and nonlinear data, which is crucial for our dataset, as it may contain complex relationships and interactions between various patient attributes and their impact on medication persistence.

```
Max SVC scores is 0.7313868613138687 for C = 100, kernel = rbf and gamma = 0.001
```



	dataset	Accuracy on test	Accuracy on train	KFold score	Balanced Accuracy	Precision 0	Precision 1	Recall 0	Recall 1
0	Original	0.731387	0.993063	0.754646	0.703277	0.772627	0.650862	0.812065	0.594488
1	Sqrt	0.881752	1.000000	0.872947	0.868037	0.894144	0.858921	0.921114	0.814961
2	Log	0.891971	1.000000	0.878787	0.879391	0.902935	0.871901	0.928074	0.830709

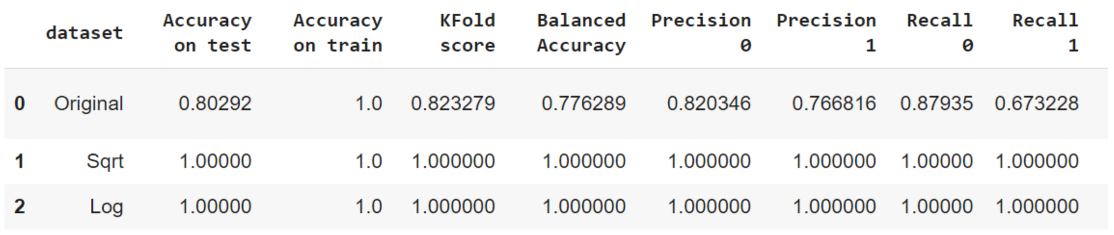


F1 score 0, 1	F1 micro	F1 weighted	ROC AUC
[0.7918552036199095, 0.6213991769547326]	0.731387	0.728650	0.703277
[0.9074285714285714, 0.8363636363636363]	0.881752	0.881077	0.868037
[0.9153318077803203, 0.8508064516129031]	0.891971	0.891406	0.879391

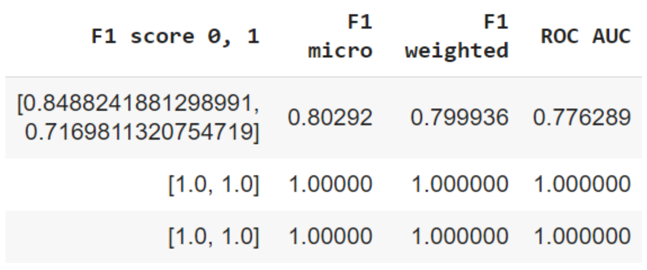
- *The Random Forest Classifier*

  The Random Forest Classifier was a pivotal choice among the models we implemented for several compelling reasons.

  First and foremost, our dataset consists of a diverse set of features, ranging from patient demographics to clinical factors and physician attributes. The Random Forest algorithm is exceptionally well-suited for our dataset, as it excels in handling mixed data types, including numerical and categorical variables. Given the presence of features such as patient age, gender, race, physician specialty, and more, this versatility makes Random Forest a robust candidate for our problem statement.

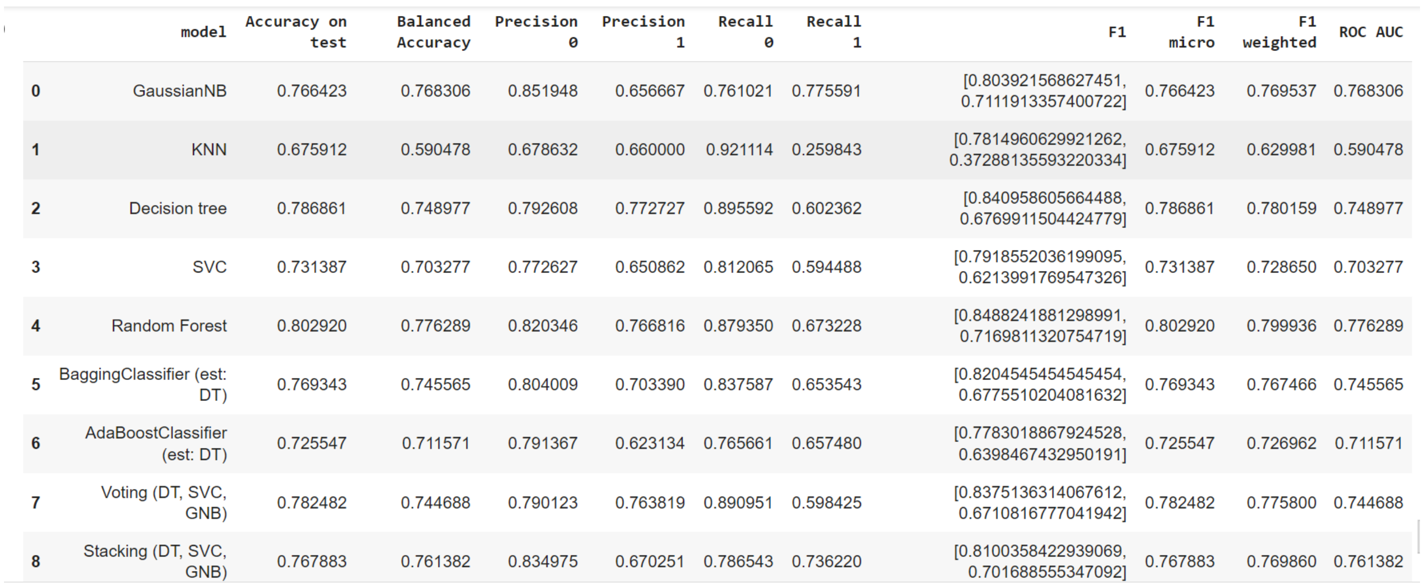


	dataset	Accuracy on test	Accuracy on train	KFold score	Balanced Accuracy	Precision 0	Precision 1	Recall 0	Recall 1
0	Original	0.80292	1.0	0.823279	0.776289	0.820346	0.766816	0.87935	0.673228
1	Sqrt	1.00000	1.0	1.000000	1.000000	1.000000	1.000000	1.00000	1.000000
2	Log	1.00000	1.0	1.000000	1.000000	1.000000	1.000000	1.00000	1.000000

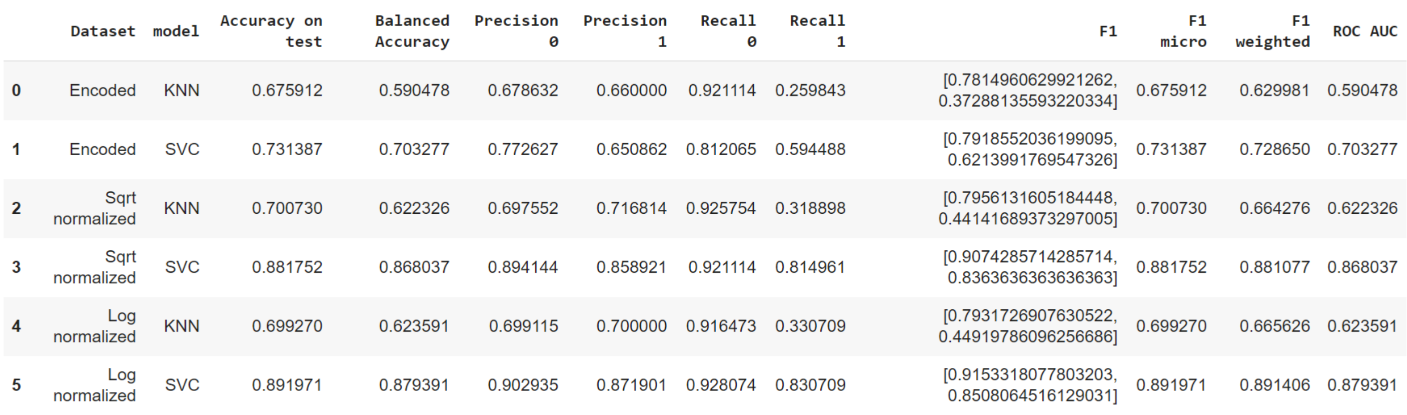


F1 score 0, 1	F1 micro	F1 weighted	ROC AUC
[0.8488241881298991, 0.7169811320754719]	0.80292	0.799936	0.776289
[1.0, 1.0]	1.00000	1.000000	1.000000
[1.0, 1.0]	1.00000	1.000000	1.000000

We implemented a few more relevant models in order to compare their performances.



	model	Accuracy on test	Balanced Accuracy	Precision 0	Precision 1	Recall 0	Recall 1	F1	F1 micro	F1 weighted	ROC AUC
0	GaussianNB	0.766423	0.768306	0.851948	0.656667	0.761021	0.775591	[0.803921568627451, 0.7111913357400722]	0.766423	0.769537	0.768306
1	KNN	0.675912	0.590478	0.678632	0.660000	0.921114	0.259843	[0.7814960629921262, 0.37288135593220334]	0.675912	0.629981	0.590478
2	Decision tree	0.786861	0.748977	0.792608	0.772727	0.895592	0.602362	[0.840958605664488, 0.6769911504424779]	0.786861	0.780159	0.748977
3	SVC	0.731387	0.703277	0.772627	0.650862	0.812065	0.594488	[0.7918552036199095, 0.6213991769547326]	0.731387	0.728650	0.703277
4	Random Forest	0.802920	0.776289	0.820346	0.766816	0.879350	0.673228	[0.8488241881298991, 0.716981132075471 9]	0.802920	0.799936	0.776289
5	BaggingClassifier (est: DT)	0.769343	0.745565	0.804009	0.703390	0.837587	0.653543	[0.8204545454545454, 0.6775510204081632]	0.769343	0.767466	0.745565
6	AdaBoostClassifier (est: DT)	0.725547	0.711571	0.791367	0.623134	0.765661	0.657480	[0.7783018867924528, 0.6398467432950191]	0.725547	0.726962	0.711571
7	Voting (DT, SVC, GNB)	0.782482	0.744688	0.790123	0.763819	0.890951	0.598425	[0.8375136314067612, 0.6710816777041942]	0.782482	0.775800	0.744688
8	Stacking (DT, SVC, GNB)	0.767883	0.761382	0.834975	0.670251	0.786543	0.736220	[0.8100358422939069, 0.701688555347092]	0.767883	0.769860	0.761382



	Dataset	model	Accuracy on test	Balanced Accuracy	Precision 0	Precision 1	Recall 0	Recall 1	F1	F1 micro	F1 weighted	ROC AUC
0	Encoded	KNN	0.675912	0.590478	0.678632	0.660000	0.921114	0.259843	[0.7814960629921262, 0.37288135593220334]	0.675912	0.629981	0.590478
1	Encoded	SVC	0.731387	0.703277	0.772627	0.650862	0.812065	0.594488	[0.7918552036199095, 0.6213991769547326]	0.731387	0.728650	0.703277
2	Sqrt normalized	KNN	0.700730	0.622326	0.697552	0.716814	0.925754	0.318898	[0.7956131605184448, 0.44141689373297005]	0.700730	0.664276	0.622326
3	Sqrt normalized	SVC	0.881752	0.868037	0.894144	0.858921	0.921114	0.814961	[0.9074285714285714, 0.8363636363636363]	0.881752	0.881077	0.868037
4	Log normalized	KNN	0.699270	0.623591	0.699115	0.700000	0.916473	0.330709	[0.7931726907630522, 0.44919786096256686]	0.699270	0.665626	0.623591
5	Log normalized	SVC	0.891971	0.879391	0.902935	0.871901	0.928074	0.830709	[0.9153318077803203, 0.8508064516129031]	0.891971	0.891406	0.879391

<u>**Conclusion: The Optimal Model and Solution**</u>

Support Vector Classifier (SVC) was revealed to be the best performing model and the optimal solution for our problem statement.

In a dataset with a class distribution of 62% for class "Non-Persistent" (0) and 38% for class "Persistent" (1). The evaluation metrics for different models are provided. For the original dataset, the SVC model had the highest scores:

- Accuracy on test: 73.14%
- Balanced Accuracy: 70.33%
- Precision for Class 0: 77.26%
- Precision for Class 1: 65.08%
- Recall for Class 0: 81.21%
- Recall for Class 1: 59.45%
- F1-Score: 71.81%

- F1-Micro: 73.14%
- F1-Weighted: 72.87%
- ROC AUC: 70.33%

The model achieves an accuracy of 73.14%, which seems reasonable. However, given the class imbalance (62% vs. 38%), accuracy alone may not provide the complete picture.
The model achieves a balanced accuracy of 70.33%, which provides a more balanced view of the model's performance.

The model correctly predicts non-persistence around 77.26% of the time, while it shows that it is correct around 65.08% of the time when it predicts persistence.

The recall for Class 0 is 81.21%, meaning that the model correctly identifies about 81.21% of the actual non-persistent cases. The recall for Class 1 is 59.45%, indicating that the model captures about 59.45% of the actual persistent cases.

The F1-Score, which balances precision and recall, is 71.81%. F1-Micro and F1-Weighted consider class imbalances. F1-Micro is 73.14%, and F1-Weighted is 72.87%, suggesting that the model performs well on both imbalanced classes.

The ROC AUC score of 70.33% is decent. It indicates that the model is reasonably good at distinguishing between the two classes.

After Log normalization of the dataset, the SVC model improves its performance significantly. The scores of SVC model on log normalized dataset is as follows:

- Accuracy on test: 89.19%
- Balanced Accuracy: 87.94%
- Precision for Class 0: 90.29%
- Precision for Class 1: 87.19%
- Recall for Class 0: 92.81%
- Recall for Class 1: 83.07%
- F1-Score: 88.30%
- F1-Micro: 89.19%
- F1-Weighted: 89.14%
- ROC AUC: 87.94%

Log normalization helps:

- To mitigate the impact of extreme values and make the data more symmetric, which reduces the impact of outliers in the features.

- To standardize the range of values across different features, which ensures that no single feature dominates the learning process.

- To bring data closer to a normal distribution.

- To find a better decision boundary or hyperplane in the feature space, which leads to improved separation between the classes, resulting in higher precision, recall, and F1-Scores.

- To better generalization and predictive performance due to the reduction of the impact of extreme value and improving the balance of the distribution of the data.

- To reduce overfitting, by making the data more stable and consistent.

Considering that the class that predicts the patients that do or do not follow their therapy is of utmost importance, since, prediction of the patients that would potentially not follow the therapy, can enable specialists to apply a different approach and follow these patients closely to make sure they follow the therapy and can avoid further complications in their health. SVC, after log normalization improves the prediction of this class of patients significantly, hence it can be considered as a valid model.