Final Project Report

Name - Parth Shah
Email - shahp7210@gmail.com
Country - India
Specialization - Data Science

Table of Contents

Problem Description

ABC is a pharmaceutical company that wants to understand the persistency of a drug as per the physician's prescription for a patient. This company has approached an Analytics company to automate this process of identification. This Analytics company has given responsibility to Team SAAN and has asked to come up with a solution to automate the persistency of a drug for the client ABC.

Business Understanding

The pharma company ABC wants to understand about the persistency of a drug for a patient. There are a bunch of Non-Tuberculous Mycobacterial (NTM) infection data. ABC company wants to know whether a patient is persistent or not depending on the prescription data. Depending on the persistency count, ABC pharma company would produce medicines in that quantity so that they can run their business strategically.

Dataset

Bucket	Variable	Variable Description
Unique Row Id	Patient ID	Unique ID of each patient
Target Variable	Persistency_Flag	Flag indicating if a patient was persistent or not
Demographics	Age	Age of the patient during their therapy
	Race	Race of the patient from the patient table
	Region	Region of the patient from the patient table
	Ethnicity	Ethnicity of the patient from the patient table
	Gender	Gender of the patient from the patient table
	IDN Indicator	Flag indicating patients mapped to IDN
Provider Attributes	NTM - Physician Specialty	Specialty of the HCP that prescribed the NTM Rx
Clinical Factors	NTM - T-Score	T Score of the patient at the time of the NTM Rx (within 2 years prior from rxdate)
	Change in T Score	Change in Tscore before starting with any therapy and after receiving therapy (Worsened, Remained Same, Improved, Unknown)
	NTM - Risk Segment	Risk Segment of the patient at the time of the NTM Rx (within 2 years days prior from rxdate)
	Change in Risk Segment	Change in Risk Segment before starting with any therapy and after receiving therapy (Worsened, Remained Same, Improved, Unknown)
	NTM - Multiple Risk Factors	Flag indicating if patient falls under multiple risk category (having more than 1 risk) at the time of the NTM Rx (within 365 days prior from rxdate)
	NTM - Dexa Scan Frequency	Number of DEXA scans taken prior to the first NTM Rx date (within 365 days prior from rxdate)
	NTM - Dexa Scan Recency	Flag indicating the presence of Dexa Scan before the NTM Rx (within 2 years prior from rxdate or between their first Rx and Switched Rx; whichever is smaller and applicable)
Disease/Treatment Factor	Dexa During Therapy	Flag indicating if the patient had a Dexa Scan during their first continuous therapy
	NTM - Fragility Fracture Recency	Flag indicating if the patient had a recent fragility fracture (within 365 days prior from rxdate)
	Fragility Fracture During Therapy	Flag indicating if the patient had fragility fracture during their first continuous therapy
	NTM - Glucocorticoid Recency	Flag indicating usage of Glucocorticoids (>=7.5mg strength) in the one year look-back from the first NTM Rx
	Glucocorticoid Usage During Therapy	Flag indicating if the patient had a Glucocorticoid usage during the first continuous therapy
	NTM - Injectable Experience	Flag indicating any injectable drug usage in the recent 12 months before the NTM OP Rx
	NTM - Risk Factors	Risk Factors that the patient is falling into. For chronic Risk Factors complete lookback to be applied and for non-chronic Risk Factors, one year lookback from the date of first OP Rx
	NTM - Comorbidity	Comorbidities are divided into two main categories - Acute and chronic, based on the ICD codes. For chronic disease we are taking complete look back from the first Rx date of NTM therapy and for acute diseases, time period before the NTM OP Rx with one year lookback has been applied
	NTM - Concomitancy	Concomitant drugs recorded prior to starting with a therapy(within 365 days prior from first rxdate)
	Adherence	Adherence for the therapies

Project Lifecycle

TASKS	17th Apr Week 0	24th Apr Week 1	1st May Week 2	8th May Week 3	15th May Week 4
Week 7					
Week 8					
Week 9					
Week 10					
Week 11					
Week 12					

Data Intake Report

Name: **Healthcare – Data Science**
Report date: **25th July 2021**
Internship Batch: **LISUM01**
Version: **1.0**
Data intake by: **Parth Shah**

Tabular data details:

Total number of observations	3424
Total number of files	1
Total number of features	26
Base format of the file	.xlsx
Size of the data	898 KB

GitHub Repository:

Project Link: https://github.com/parthshah28/healthcare-datascience

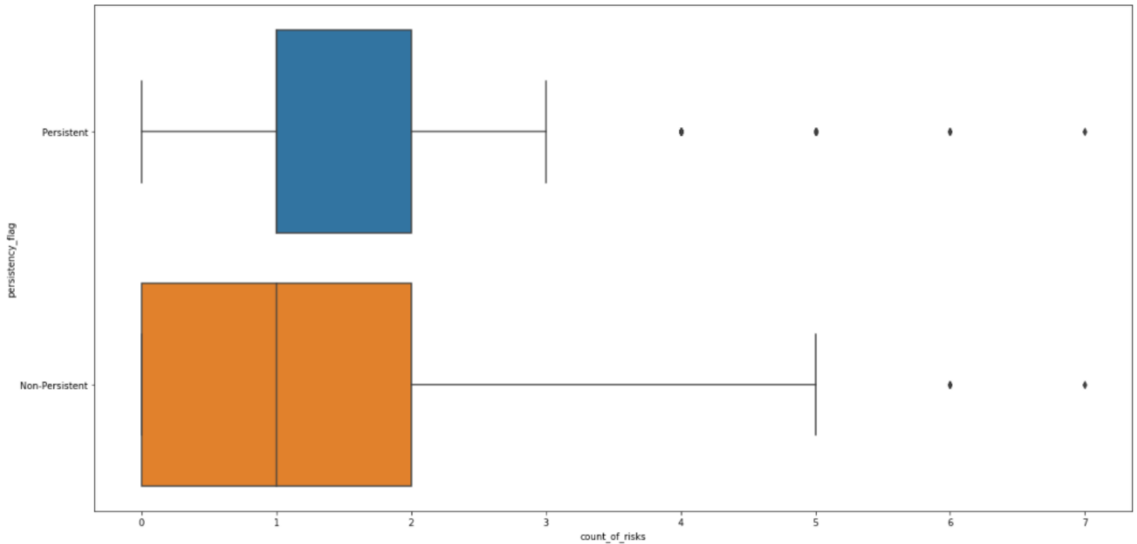
Data Types

In this dataset as you can find in data intake report, we have dataset with (3424, 69) dimension and the features that we described them with following datatypes, “object” types mean categorical columns:

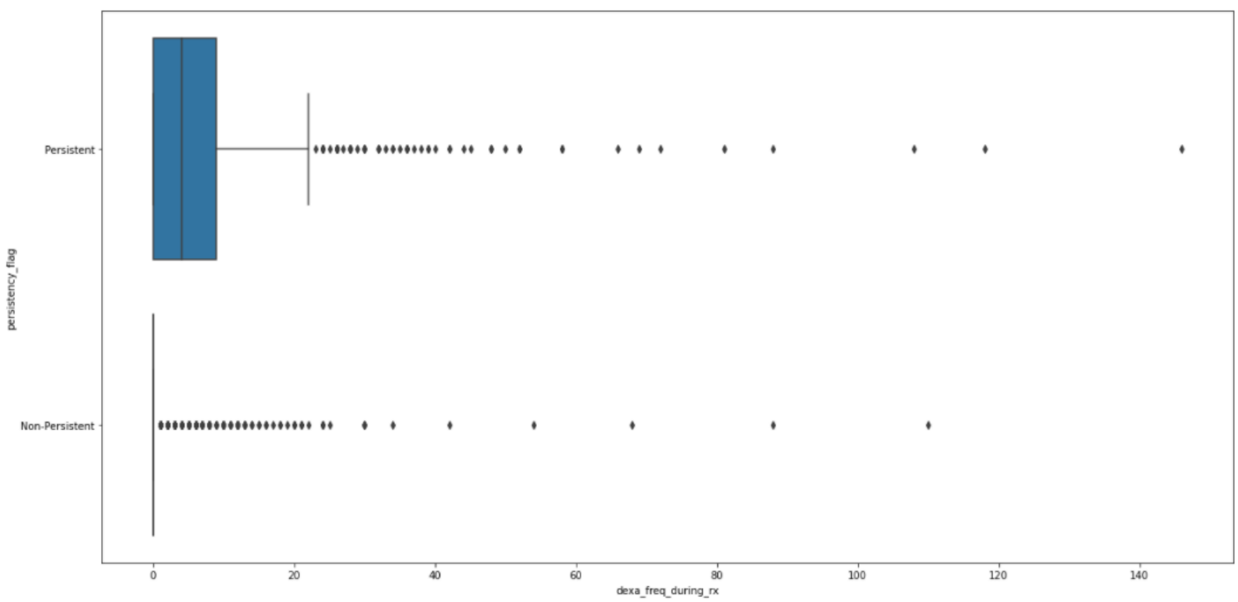
Ptid	object
Persistency_Flag	object
Gender	object
Race	object
Ethnicity	object
Region	object
Age_Bucket	object
Ntm_Speciality	object
Ntm_Specialist_Flag	object
Ntm_Speciality_Bucket	object
Gluco_Record_Prior_Ntm	object
Gluco_Record_During_Rx	object
Dexa_Freq_During_Rx	int64
Dexa_During_Rx	object
Frag_Frac_Prior_Ntm	object
Frag_Frac_During_Rx	object
Risk_Segment_Prior_Ntm	object
Tscore_Bucket_Prior_Ntm	object
Risk_Segment_During_Rx	object
Tscore_Bucket_During_Rx	object
Change_T_Score	object
Change_Risk_Segment	object
Adherent_Flag	object
Idn_Indicator	object
Injectable_Experience_During_Rx	object
Comorb_Encounter_For_Screening_For_Malignant_Neoplasms	object
Comorb_Encounter_For_Immunization	object
Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx	object
Comorb_Vitamin_D_Deficiency	object
Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified	object
Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx	object
Comorb_Long_Term_Current_Drug_Therapy	object
Comorb_Dorsalgia	object
Comorb_Personal_History_Of_Other_Diseases_And_Conditions	object
Comorb_Other_Disorders_Of_Bone_Density_And_Structure	object
Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias	object
Comorb_Osteoporosis_without_current_pathological_fracture	object
Comorb_Personal_history_of_malignant_neoplasm	object
Comorb_Gastro_esophageal_reflux_disease	object
Concom_Cholesterol_And_Triglyceride_Regulating_Preparations	object
Concom_Narcotics	object
Concom_Systemic_Corticosteroids_Plain	object
Concom_Anti_Depressants_And_Mood_Stabilisers	object
Concom_Fluoroquinolones	object
Concom_Cephalosporins	object
Concom_Macrolides_And_Similar_Types	object
Concom_Broad_Spectrum_Penicillins	object
Concom_Anaesthetics_General	object
Concom_Viral_Vaccines	object
Risk_Type_1_Insulin_Dependent_Diabetes	object
Risk_Osteogenesis_Imperfecta	object
Risk_Rheumatoid_Arthritis	object
Risk_Untreated_Chronic_Hyperthyroidism	object
Risk_Untreated_Chronic_Hypogonadism	object
Risk_Untreated_Early_Menopause	object
Risk_Patient_Parent_Fractured_Their_Hip	object
Risk_Smoking_Tobacco	object
Risk_Chronic_Malnutrition_Or_Malabsorption	object
Risk_Chronic_Liver_Disease	object
Risk_Family_History_Of_Osteoporosis	object
Risk_Low_Calcium_Intake	object
Risk_Vitamin_D_Insufficiency	object
Risk_Poor_Health_Frailty	object
Risk_Excessive_Thinness	object
Risk_Hysterectomy_Oophorectomy	object
Risk_Estrogen_Deficiency	object
Risk_Immobilization	object
Risk_Recurring_Falls	object
Count_Of_Risks	int64

Data Problems

- Null Values: This dataset has no Null values
- Outliers: We have only two numerical columns and both of them have some outliers.
  - **count_of_risks:**



  - **dexa_freq_during_rx:**



- Skewness and Kurtosis: We have only two numerical columns and both of them have some outliers.
  - count_of_risks:

    Count of risks skweness: 0.8797905232898707

    Count of risks Kurtosis: 0.9004859968892842
  - dexa_freq_during_rx:

    dexa_freq_during_rx skweness: 6.8087302112992285

    dexa_freq_during_rx Kurtosis: 74.75837754795428

Data Transformation

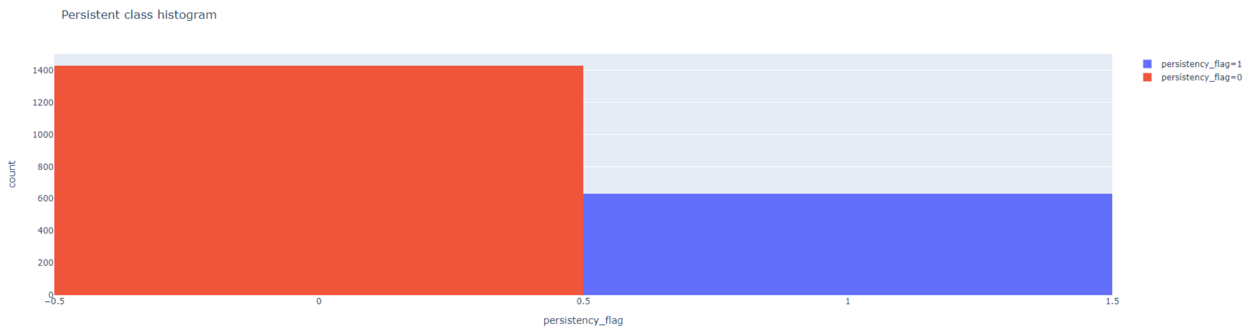
As we did not have any Null values, so we have nothing to do in this regard. We have some skewness and Kurtosis in our two numerical features, so we will scaled their values by RobustScaler() and after that remove their outliers by calculating IQR and remove data smaller/greater than two whiskers. After removing outliers from “dexa_freq_during_rx” we can check how much we have decrease in the shape of the data:

Old Shape: (3424, 69)

New Shape: (2964, 69)

We have changed all the [‘Y’, ‘N’] values to [1, 0] to train models on the data, and also we change the values of target feature in this way :
['Non-Persistent', 'Persistent'] to [0, 1].

The other thing that we had to overcome on this dataset is the unbalancing of the target feature:



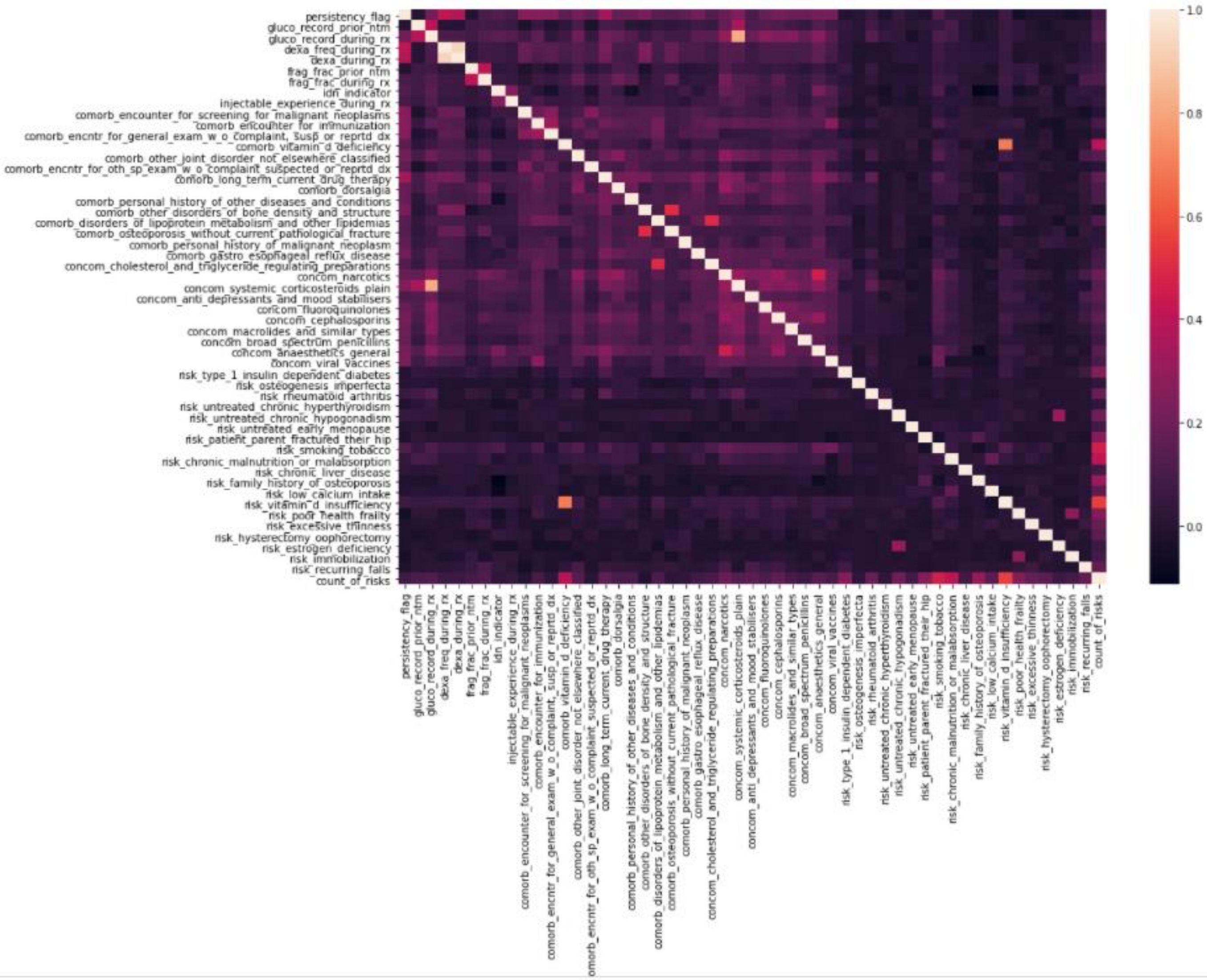
since imbalanced datasets make predicting hard and don't let models work well on them! One of good things that we can do is "Up sampling", in this method we increase the records of the minority class, at last we have same count of records of each class.

The other thing that we performed on the dataset is "one hot encoding", For using classifiers we need numerical values, to do this I used One Hot Encoding that implemented by "get_dummies()" function from Pandas library, it works like this:

ID	Gender
1	Male
2	Female
3	Not Specified
4	Not Specified
5	Female

ID	Male	Female	Not Specified
1	1	0	0
2	0	1	0
3	0	0	1
4	0	0	1
5	0	1	0

Data Dependency



Final Recommendation

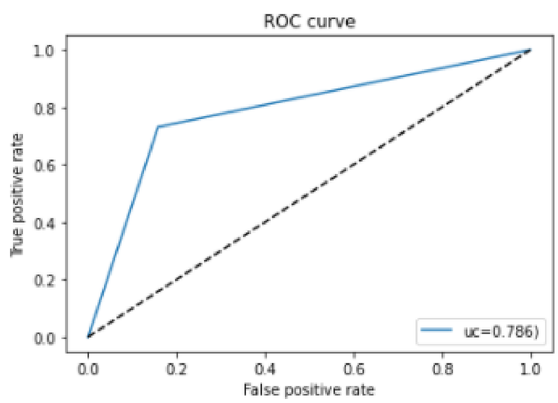
Now we can perform classifiers models on the train set which we get it by splitting whole dataset to train and test sets in the way 70% for tarin set and 30% test set.

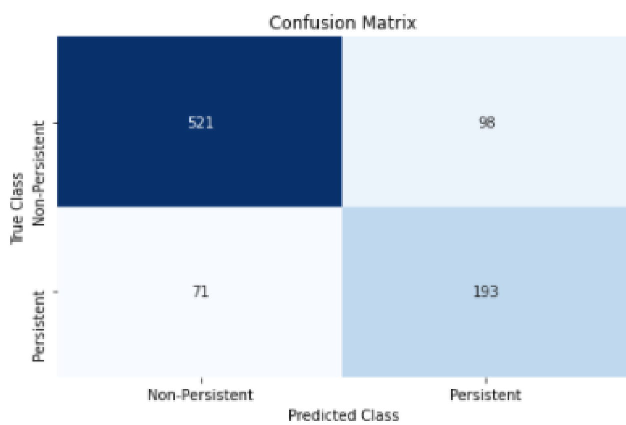
Model deployment

Here we will see results of different classification models which are linear models, ensemble and boosting models and also neural networks models:

- Linear Models
    - *LogisticRegression:*

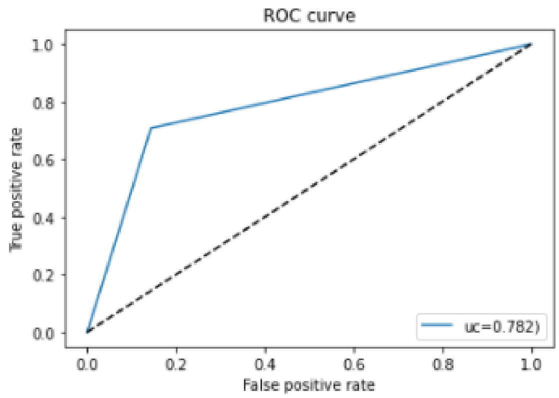
```
Accuracy : 0.8086070215175538
Precision : 0.6632302405498282
Recall : 0.7310606060606061
F1 Score : 0.6954954954954955
                precision    recall  f1-score   support

Non-Persistent       0.88      0.84      0.86       619
    Persistent       0.66      0.73      0.70       264

      accuracy                           0.81       883
     macro avg       0.77      0.79      0.78       883
  weighted avg       0.82      0.81      0.81       883

AUC : 0.7863703676506584
```

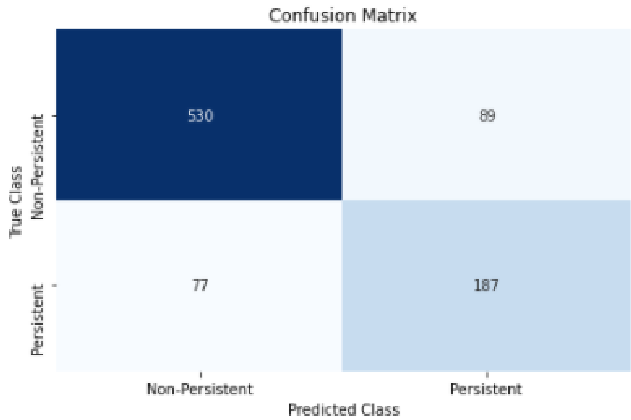




- *RidgeClassifier:*

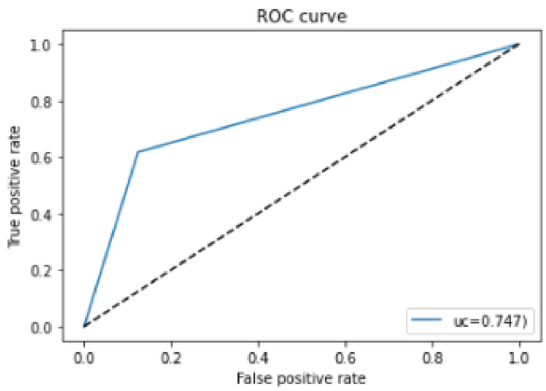
```
Accuracy : 0.812004530011325
Precision : 0.677536231884058
Recall : 0.7083333333333334
F1 Score : 0.6925925925925926
                precision    recall  f1-score   support

Non-Persistent       0.87      0.86      0.86       619
    Persistent       0.68      0.71      0.69       264

      accuracy                           0.81       883
     macro avg       0.78      0.78      0.78       883
  weighted avg       0.81      0.81      0.81       883

AUC : 0.782276521270867
```

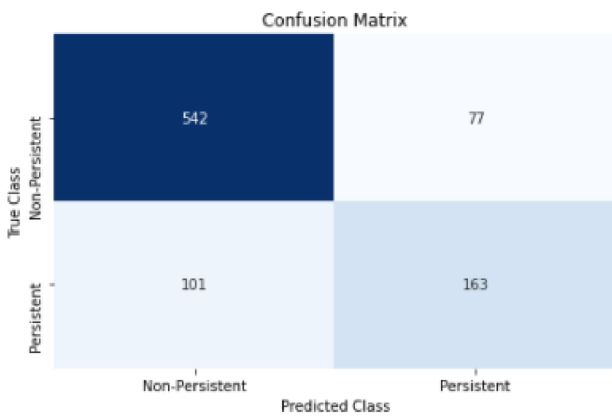




- *SGDClassifier:*

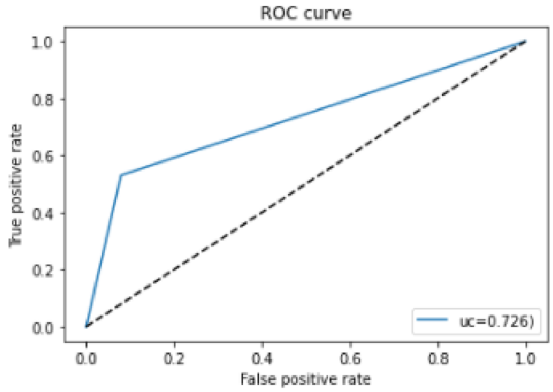
```
Accuracy : 0.79841449603624
Precision : 0.6791666666666667
Recall : 0.6174242424242424
F1 Score : 0.6468253968253969
               precision    recall  f1-score   support

Non-Persistent       0.84      0.88      0.86       619
    Persistent       0.68      0.62      0.65       264

      accuracy                           0.80       883
     macro avg       0.76      0.75      0.75       883
  weighted avg       0.79      0.80      0.80       883

AUC : 0.7465150291281147
```

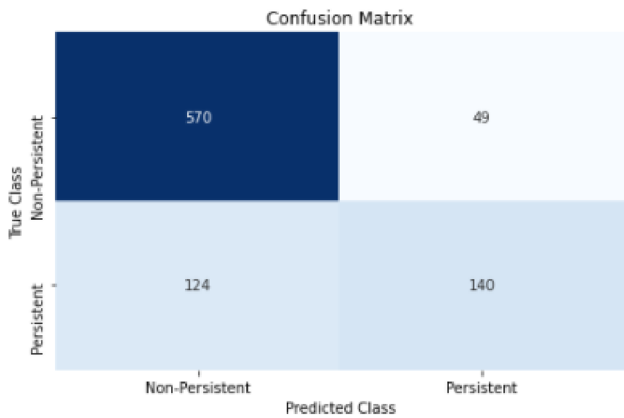




- Ensemble and Boosting Models
  - *RandomForestClassifier:*

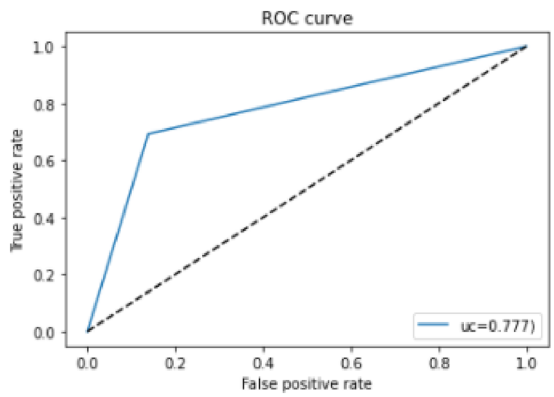
```
Accuracy : 0.8040770101925255
Precision : 0.7407407407407407
Recall : 0.5303030303030303
F1 Score : 0.6181015452538631
               precision    recall  f1-score   support

Non-Persistent       0.82      0.92      0.87       619
    Persistent       0.74      0.53      0.62       264

      accuracy                           0.80       883
     macro avg       0.78      0.73      0.74       883
  weighted avg       0.80      0.80      0.79       883

AUC : 0.7255715474616928
```

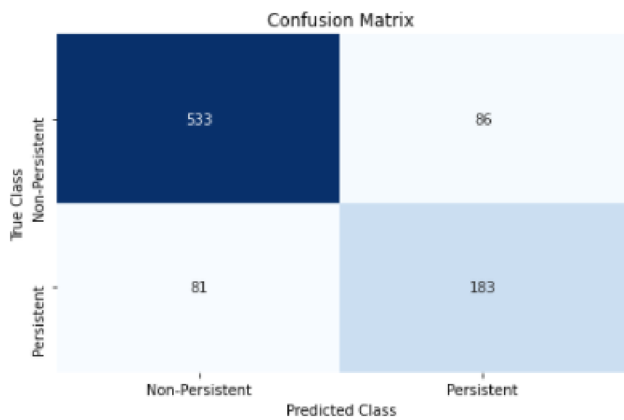




- *BaggingClassifier:*

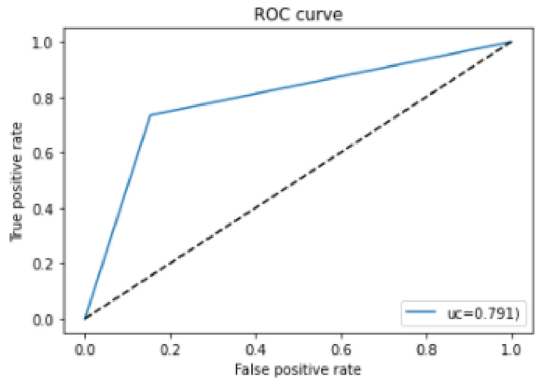
```
Accuracy : 0.8108720271800679
Precision : 0.6802973977695167
Recall : 0.6931818181818182
F1 Score : 0.6866791744840526
                precision    recall  f1-score   support

Non-Persistent       0.87      0.86      0.86       619
    Persistent       0.68      0.69      0.69       264

      accuracy                           0.81       883
     macro avg       0.77      0.78      0.78       883
  weighted avg       0.81      0.81      0.81       883

AUC : 0.7771240270230578
```

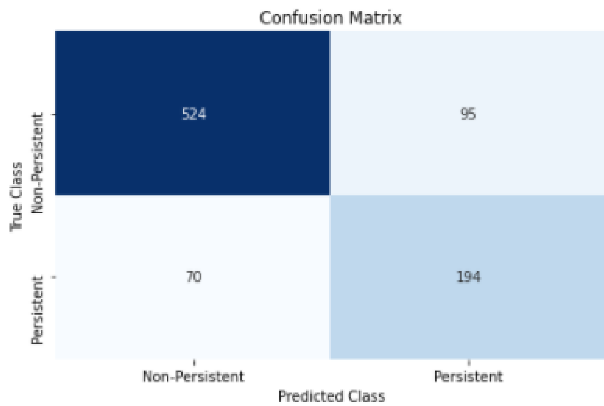




- *AdaBoostClassifier:*

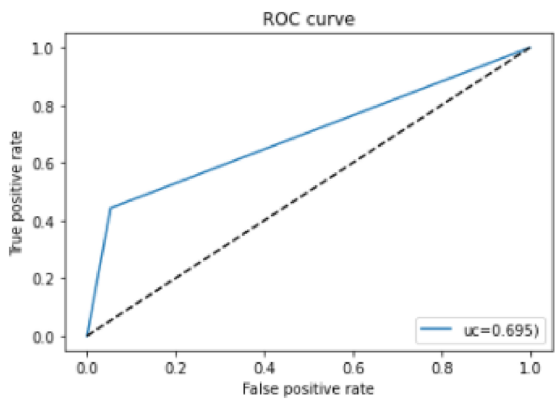
```
Accuracy : 0.8131370328425821
Precision : 0.671280276816609
Recall : 0.7348484848484849
F1 Score : 0.701627486437613
              precision    recall  f1-score   support

Non-Persistent       0.88      0.85      0.86       619
    Persistent       0.67      0.73      0.70       264

      accuracy                           0.81       883
     macro avg       0.78      0.79      0.78       883
  weighted avg       0.82      0.81      0.82       883

AUC : 0.7906875703725462
```

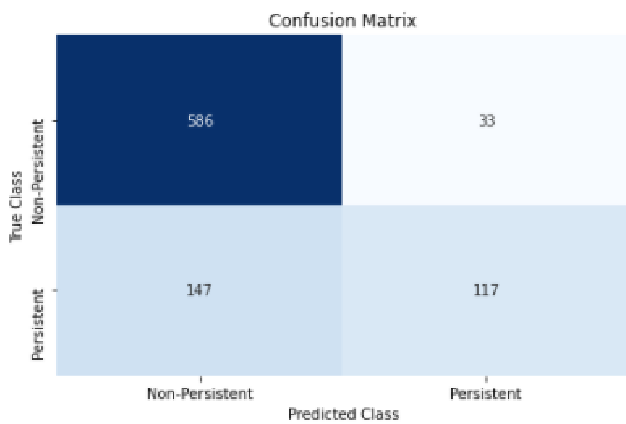




- *ExtraTreesClassifier:*

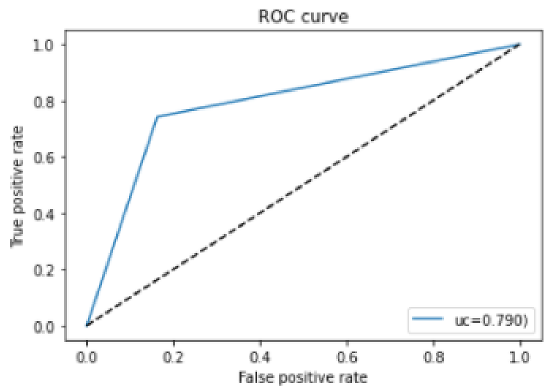
```
Accuracy : 0.796149490373726
Precision : 0.78
Recall : 0.4431818181818182
F1 Score : 0.5652173913043479
              precision    recall  f1-score   support

Non-Persistent       0.80      0.95      0.87       619
    Persistent       0.78      0.44      0.57       264

      accuracy                           0.80       883
     macro avg       0.79      0.69      0.72       883
  weighted avg       0.79      0.80      0.78       883

AUC : 0.6949350124834778
```

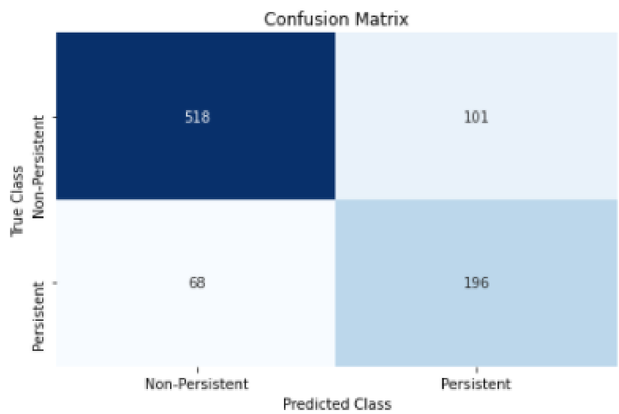




- *GradientBoostingClassifier:*

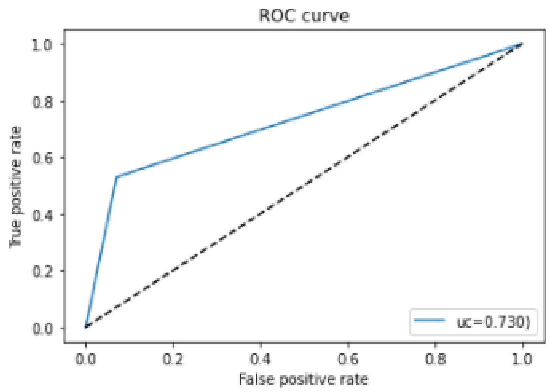
```
Accuracy : 0.8086070215175538
Precision : 0.6599326599326599
Recall : 0.7424242424242424
F1 Score : 0.698752228163993
               precision    recall  f1-score   support

Non-Persistent       0.88      0.84      0.86       619
    Persistent       0.66      0.74      0.70       264

      accuracy                           0.81       883
     macro avg       0.77      0.79      0.78       883
  weighted avg       0.82      0.81      0.81       883

AUC : 0.7896289225045283
```

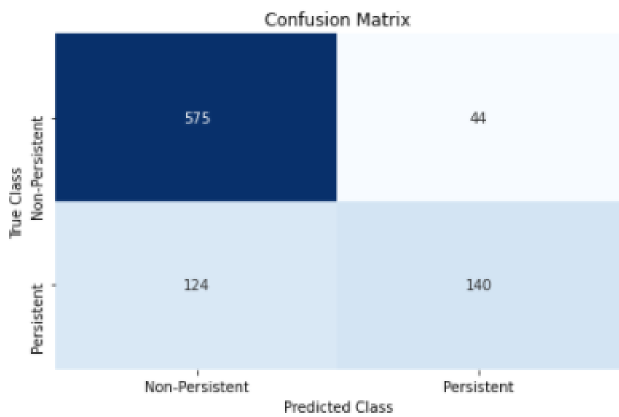




- *StackingClassifier:*

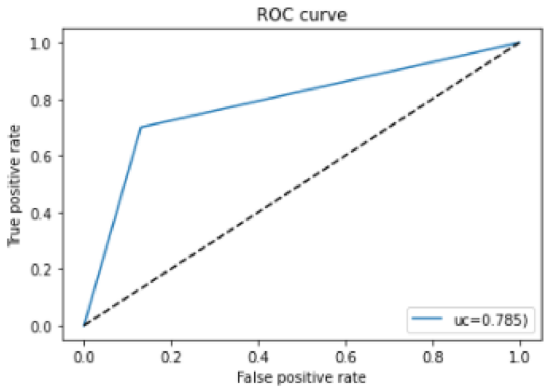
```
Accuracy : 0.8097395243488109
Precision : 0.7608695652173914
Recall : 0.5303030303030303
F1 Score : 0.625
                precision    recall  f1-score   support

Non-Persistent       0.82      0.93      0.87       619
    Persistent       0.76      0.53      0.62       264

      accuracy                           0.81       883
     macro avg       0.79      0.73      0.75       883
  weighted avg       0.80      0.81      0.80       883

AUC : 0.7296103196749399
```

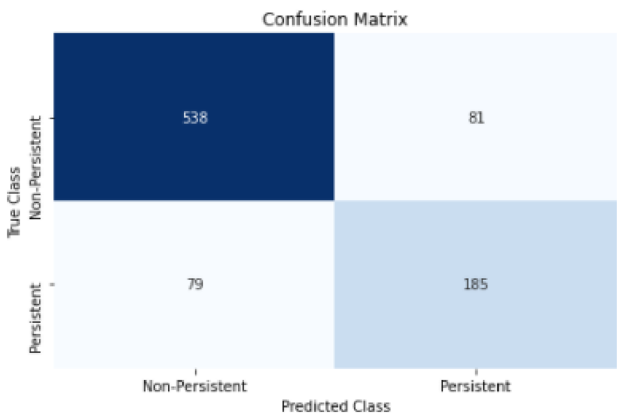




- *XGBoostClassifier:*

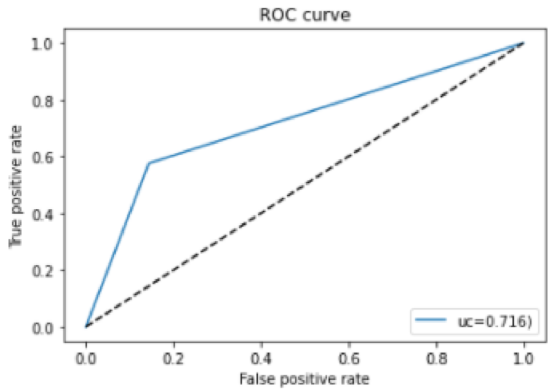
```
Accuracy : 0.8187995469988675
Precision : 0.6954887218045113
Recall : 0.7007575757575758
F1 Score : 0.6981132075471698
               precision    recall  f1-score   support

Non-Persistent       0.87      0.87      0.87       619
    Persistent       0.70      0.70      0.70       264

      accuracy                           0.82       883
     macro avg       0.78      0.78      0.78       883
  weighted avg       0.82      0.82      0.82       883

AUC : 0.7849506780241836
```

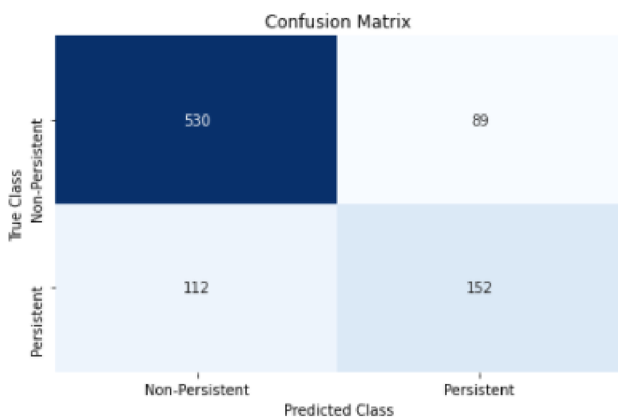




- Neural Network Models
    - *Multi-Layer Perceptron:*

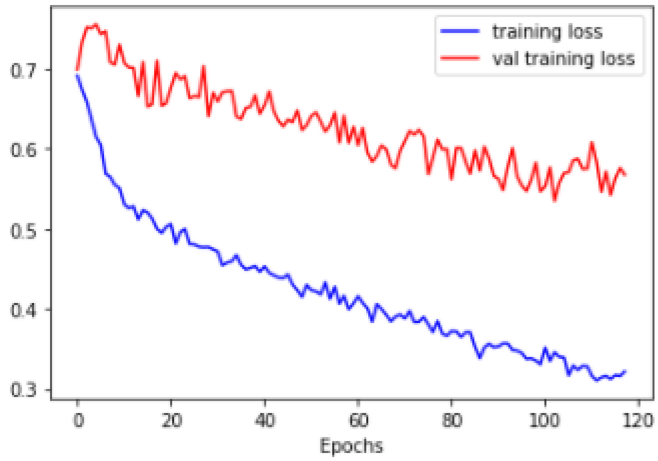
```
Accuracy : 0.7723669309173273
Precision : 0.6307053941908713
Recall : 0.5757575757575758
F1 Score : 0.6019801980198021
                precision    recall  f1-score   support

Non-Persistent       0.83      0.86      0.84       619
    Persistent       0.63      0.58      0.60       264

      accuracy                           0.77       883
     macro avg       0.73      0.72      0.72       883
  weighted avg       0.77      0.77      0.77       883

AUC : 0.7159886424829882
```

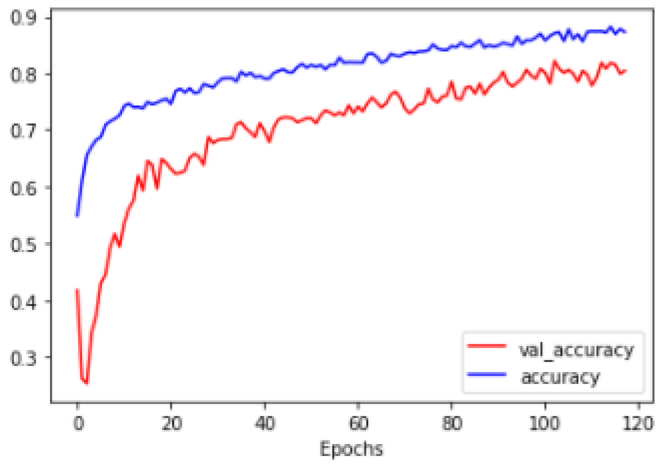




- *Multilayer Neural Network with Tensorflow/Keras:*

Epoch 00118: early stopping





Accuracy : 0.8029445073612684
Precision : 0.6875
Recall : 0.625
F1 Score : 0.6547619047619048

	precision	recall	f1-score	support
Non-Persistent	0.85	0.88	0.86	619
Persistent	0.69	0.62	0.65	264
accuracy			0.80	883
macro avg	0.77	0.75	0.76	883
weighted avg	0.80	0.80	0.80	883

AUC : 0.7519184168012925

Conclusion

Approximately all the classifiers have same result, but three of them are the bests and their result are so close to each other:

- RidgeClassifier (Linear)
- AdaBoostClassifier (Ensemble/Boosting)
- XGBoostClassifier (Ensemble/Boosting)

They have around 81% Accuracy, 68% Precision, 71% Recall, 70% F1 Score, 78% AUC. We can also see the results for each classifier as well.

Training Final Model

As we said in last part, all the model have Approximately save results so we need one of them, for example StackingClassifier and deploy it on whole dataset and save it to final_model.sav.