Imports

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from scipy import stats
import collections
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
from IPython.core.interactiveshell import InteractiveShell
import warnings

#importing packages for modeling
from sklearn.linear_model import LogisticRegression, RidgeClassifier, SGDClassifie
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, AdaBoostCla
from xgboost import XGBClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import make_pipeline

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.layers import Dropout
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import BatchNormalization

%matplotlib inline
InteractiveShell.ast_node_interactivity = "all"
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
warnings.filterwarnings('ignore')
```

```
target_names=['Non-Persistent', 'Persistent']
```

Functions

```
def evaluation_metrics(y_test, y_pre, target_names):
    #scores
    print("Accuracy :",accuracy_score(y_test,y_pre))
    print("Precision :",precision_score(y_test,y_pre))
```

```
    print("Precision :",precision_score(y_test,y_pre))
    print("Recall :",recall_score(y_test,y_pre))
    print("F1 Score :",f1_score(y_test,y_pre))

    print(classification_report(y_test, y_pre, target_names=target_names))

    #AUC
    fpr, tpr, _ = roc_curve(y_test,  y_pre)
    auc = roc_auc_score(y_test, y_pre)
    print("AUC :", auc)

    #ROC
    plt.plot(fpr,tpr,label="uc={:.3f})".format(auc))
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.title('ROC curve')
    plt.legend(loc=4)
    plt.show()

    #CM matrix
    matrix = confusion_matrix(y_test, y_pre)
    cm = pd.DataFrame(matrix, index=target_names, columns=target_names)

    sns.heatmap(cm, annot=True, cbar=None, cmap="Blues", fmt = 'g')
    plt.title("Confusion Matrix"), plt.tight_layout()
    plt.ylabel("True Class"), plt.xlabel("Predicted Class")
    plt.show()
```

```
def logistic(X_train,X_test,y_train,y_test):
    model=LogisticRegression()
    model.fit(X_train,y_train)
    y_pre=model.predict(X_test)
    evaluation_metrics(y_test, y_pre, target_names)
```

```
def Ridge(X_train,X_test,y_train,y_test):
    #train the model
    model = RidgeClassifier(random_state=2)
    model.fit(X_train, y_train)
    #predictions
    y_pre = model.predict(X_test)
    evaluation_metrics(y_test, y_pre, target_names)
```

```
def SGD(X_train,X_test,y_train,y_test):
    #train the model
    model = SGDClassifier()
    model.fit(X_train, y_train)
    #predictions
    y_pre = model.predict(X_test)
    evaluation_metrics(y_test, y_pre, target_names)
```

```
def XGBOOST(X_train,X_test,y_train,y_test):
```

```
  #train the model
  model = XGBClassifier(random_state=2)
  model.fit(X_train, y_train)
  #predictions
  y_pre = model.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

```
def RF(X_train,X_test,y_train,y_test):
  #train the model
  model = RandomForestClassifier(random_state=2)
  model.fit(X_train, y_train)
  #predictions
  y_pre = model.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

```
def Bagging(X_train,X_test,y_train,y_test):
  #train the model
  model = BaggingClassifier(base_estimator=SVC(), n_estimators=10, random_state=0)
  model.fit(X_train, y_train)
  #predictions
  y_pre = model.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

```
def AdaBoost(X_train,X_test,y_train,y_test):
  #train the model
  model = AdaBoostClassifier(n_estimators=100, random_state=0)
  model.fit(X_train, y_train)
  #predictions
  y_pre = model.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

```
def ExtraTrees(X_train,X_test,y_train,y_test):
  #train the model
  model = ExtraTreesClassifier(n_estimators=100, random_state=0)
  model.fit(X_train, y_train)
  #predictions
  y_pre = model.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

```
def GradientBoosting(X_train,X_test,y_train,y_test):
  #train the model
  model = GradientBoostingClassifier(n_estimators = 600, max_depth = 20, min_sampl
  model.fit(X_train, y_train)
  #predictions
  y_pre = model.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

```
def Stacking(X_train,X_test,y_train,y_test):
  #train the model
  estimators = [('rf', RandomForestClassifier(n_estimators=10, random_state=42)),
  model = StackingClassifier(estimators=estimators, final_estimator=LogisticRegres
```

```
  model.fit(X_train, y_train)
  #predictions
  y_pre = model.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

```
def MLP(X_train,X_test,y_train,y_test):
  #train the model
  mlp = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(100,100), random_state=2
  mlp.fit(X_train, y_train)
  mlp.get_params(deep=True)
  #predictions
  y_pre = mlp.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

```
def MNN(X_train,X_test,y_train,y_test):
  #train the model
  model = Sequential()
  model.add(Dense(32, input_shape=(X_train.shape[1],), activation='relu')),
  model.add(Dropout(0.2)),
  model.add(Dense(16, activation='relu')),
  model.add(Dropout(0.2)),
  model.add(Dense(8, activation='relu')),
  model.add(Dropout(0.2)),
  model.add(Dense(4, activation='relu')),
  model.add(Dropout(0.2)),
  model.add(Dense(1, activation='sigmoid'))

  opt = tf.keras.optimizers.Adam(learning_rate=0.0001) #optimizer

  model.compile(optimizer=opt, loss=tf.keras.losses.BinaryCrossentropy(), metrics=

  earlystopper = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', min_delta

  history = model.fit(X_train.values, y_train.values, epochs = 150, batch_size=10,
  history_dict = history.history

  loss_values = history_dict['loss']
  val_loss_values=history_dict['val_loss']
  plt.plot(loss_values,'b',label='training loss')
  plt.plot(val_loss_values,'r',label='val training loss')
  plt.legend()
  plt.xlabel("Epochs")
  plt.show()

  accuracy_values = history_dict['accuracy']
  val_accuracy_values=history_dict['val_accuracy']
  plt.plot(val_accuracy_values,'-r',label='val_accuracy')
  plt.plot(accuracy_values,'-b',label='accuracy')
  plt.legend()
  plt.xlabel("Epochs")
  plt.show()

  #predictions
```

```
y_pre = model.predict_classes(X_test)

evaluation_metrics(y_test, y_pre, target_names)
```

Reading data

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

```
!ls drive/MyDrive/Project-DG-2021/Dataset
```

```
Healthcare_dataset.xlsx
```

```
xls = pd.ExcelFile('drive/MyDrive/Project-DG-2021/Dataset/Healthcare_dataset.xlsx'
df= pd.read_excel(xls, 'Dataset')
```

Data Understanding

```
df.head()
```

	Ptid	Persistency_Flag	Gender	Race	Ethnicity	Region	Age_Bucket
0	P1	Persistent	Male	Caucasian	Not Hispanic	West	>75
1	P2	Non-Persistent	Male	Asian	Not Hispanic	West	55-65
2	P3	Non-Persistent	Female	Other/Unknown	Hispanic	Midwest	65-75
3	P4	Non-Persistent	Female	Caucasian	Not Hispanic	Midwest	>75
4	P5	Non-Persistent	Female	Caucasian	Not Hispanic	Midwest	>75

```
df.shape
```

```
(3424, 69)
```

```
df.dtypes
```

```
Gluco_Record_Prior_Ntm                                             objec
Gluco_Record_During_Rx                                             objec
Dexa_Freq_During_Rx                                                 int6
```

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

Risk_Excessive_Thinness                                             objec
Risk_Hysterectomy_Oophorectomy                                      objec
Risk_Estrogen_Deficiency                                            objec
Risk_Immobilization                                                 objec
Risk_Recurring_Falls                                                objec
Count_Of_Risks                                                       int6
dtype: object
```

```
df.info()
```

```
 12   Dexa_Freq_During_Rx                                                34
```

```
 13  Dexa_During_Rx                                                      34
 14  Frag_Frac_Prior_Ntm                                                 34
 15  Frag_Frac_During_Rx                                                 34
 16  Risk_Segment_Prior_Ntm                                              34
 17  Tscore_Bucket_Prior_Ntm                                             34
 18  Risk_Segment_During_Rx                                              34
 19  Tscore_Bucket_During_Rx                                             34
 20  Change_T_Score                                                      34
 21  Change_Risk_Segment                                                 34
 22  Adherent_Flag                                                       34
 23  Idn_Indicator                                                       34
 24  Injectable_Experience_During_Rx                                     34
 25  Comorb_Encounter_For_Screening_For_Malignant_Neoplasms              34
 26  Comorb_Encounter_For_Immunization                                   34
 27  Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx     34
 28  Comorb_Vitamin_D_Deficiency                                         34
 29  Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified                34
 30  Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx  34
 31  Comorb_Long_Term_Current_Drug_Therapy                               34
 32  Comorb_Dorsalgia                                                    34
 33  Comorb_Personal_History_Of_Other_Diseases_And_Conditions            34
 34  Comorb_Other_Disorders_Of_Bone_Density_And_Structure                34
 35  Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias     34
 36  Comorb_Osteoporosis_without_current_pathological_fracture           34
 37  Comorb_Personal_history_of_malignant_neoplasm                       34
 38  Comorb_Gastro_esophageal_reflux_disease                             34
 39  Concom_Cholesterol_And_Triglyceride_Regulating_Preparations         34
 40  Concom_Narcotics                                                    34
 41  Concom_Systemic_Corticosteroids_Plain                               34
 42  Concom_Anti_Depressants_And_Mood_Stabilisers                        34
 43  Concom_Fluoroquinolones                                             34
 44  Concom_Cephalosporins                                               34
 45  Concom_Macrolides_And_Similar_Types                                 34
 46  Concom_Broad_Spectrum_Penicillins                                   34
 47  Concom_Anaesthetics_General                                         34
 48  Concom_Viral_Vaccines                                               34
 49  Risk_Type_1_Insulin_Dependent_Diabetes                              34
 50  Risk_Osteogenesis_Imperfecta                                        34
 51  Risk_Rheumatoid_Arthritis                                           34
 52  Risk_Untreated_Chronic_Hyperthyroidism                              34
 53  Risk_Untreated_Chronic_Hypogonadism                                 34
 54  Risk_Untreated_Early_Menopause                                      34
 55  Risk_Patient_Parent_Fractured_Their_Hip                             34
 56  Risk_Smoking_Tobacco                                                34
 57  Risk_Chronic_Malnutrition_Or_Malabsorption                          34
 58  Risk_Chronic_Liver_Disease                                          34
 59  Risk_Family_History_Of_Osteoporosis                                 34
 60  Risk_Low_Calcium_Intake                                             34
 61  Risk_Vitamin_D_Insufficiency                                        34
 62  Risk_Poor_Health_Frailty                                            34
 63  Risk_Excessive_Thinness                                             34
 64  Risk_Hysterectomy_Oophorectomy                                      34
 65  Risk_Estrogen_Deficiency                                            34
 66  Risk_Immobilization                                                 34
 67  Risk_Recurring_Falls                                                34
 68  Count_Of_Risks                                                      34
dtypes: int64(2), object(67)
```

```
df.columns=[x.lower() for x in df.columns]
```

Analyzing dependency of variable (Before Transformation)

```
classes=df['persistency_flag'].value_counts()
normal_share=round(classes[0]/df['persistency_flag'].count()*100,2)
fraud_share=round(classes[1]/df['persistency_flag'].count()*100, 2)
print("Non-Persistent : {} %".format(normal_share))
print("Persistent : {} %".format(fraud_share))
```

```
Non-Persistent : 62.35 %
Persistent : 37.65 %
```

```
cat_corr = df.apply(lambda x : pd.factorize(x)[0]).corr(method='pearson', min_peri
np.abs(cat_corr).sort_values(by=['persistency_flag'], ascending=False)
```

	persistency_fla
persistency_flag	1.00000
dexa_during_rx	0.49182
dexa_freq_during_rx	0.39524
comorb_long_term_current_drug_therapy	0.35276
comorb_encounter_for_screening_for_malignant_neoplasms	0.32232
comorb_encounter_for_immunization	0.31488
comorb_encntr_for_general_exam_w_o_complaint,_susp_or_reprtd_dx	0.28982
comorb_other_disorders_of_bone_density_and_structure	0.24728
concom_systemic_corticosteroids_plain	0.24285
comorb_other_joint_disorder_not_elsewhere_classified	0.23327
concom_anaesthetics_general	0.22229
concom_viral_vaccines	0.22224
concom_macrolides_and_similar_types	0.2216
concom_cephalosporins	0.22154
comorb_gastro_esophageal_reflux_disease	0.22064
comorb_personal_history_of_other_diseases_and_conditions	0.21966
comorb_dorsalgia	0.21530
comorb_encntr_for_oth_sp_exam_w_o_complaint_suspected_or_reprtd_dx	0.21341
gluco_record_during_rx	0.21270
concom_broad_spectrum_penicillins	0.19785
concom_narcotics	0.19191
concom_fluoroquinolones	0.18619
comorb_personal_history_of_malignant_neoplasm	0.17483
comorb_vitamin_d_deficiency	0.17266
comorb_disorders_of_lipoprotein_metabolism_and_other_lipidemias	0.16349
comorb_osteoporosis_without_current_pathological_fracture	0.13992
ntm_specialist_flag	0.13938
concom_cholesterol_and_triglyceride_regulating_preparations	0.12555
adherent_flag	0.11248
idn_indicator	0.11144
concom_anti_depressants_and_mood_stabilisers	0.11004
frag_frac_during_rx	0.10693
change_risk_segment	0.10619

change_risk_segment	0.1061
injectable_experience_during_rx	0.0983
risk_smoking_tobacco	0.0980
ntm_speciality_bucket	0.0916
risk_vitamin_d_insufficiency	0.0797
count_of_risks	0.0715
risk_untreated_chronic_hypogonadism	0.0675
risk_rheumatoid_arthritis	0.0538
risk_immobilization	0.0497
risk_chronic_malnutrition_or_malabsorption	0.0491
risk_poor_health_frailty	0.0452
risk_excessive_thinness	0.0401
change_t_score	0.0230
ethnicity	0.0225
ptid	0.0200

Missing Values

```
df.isnull().sum()
```

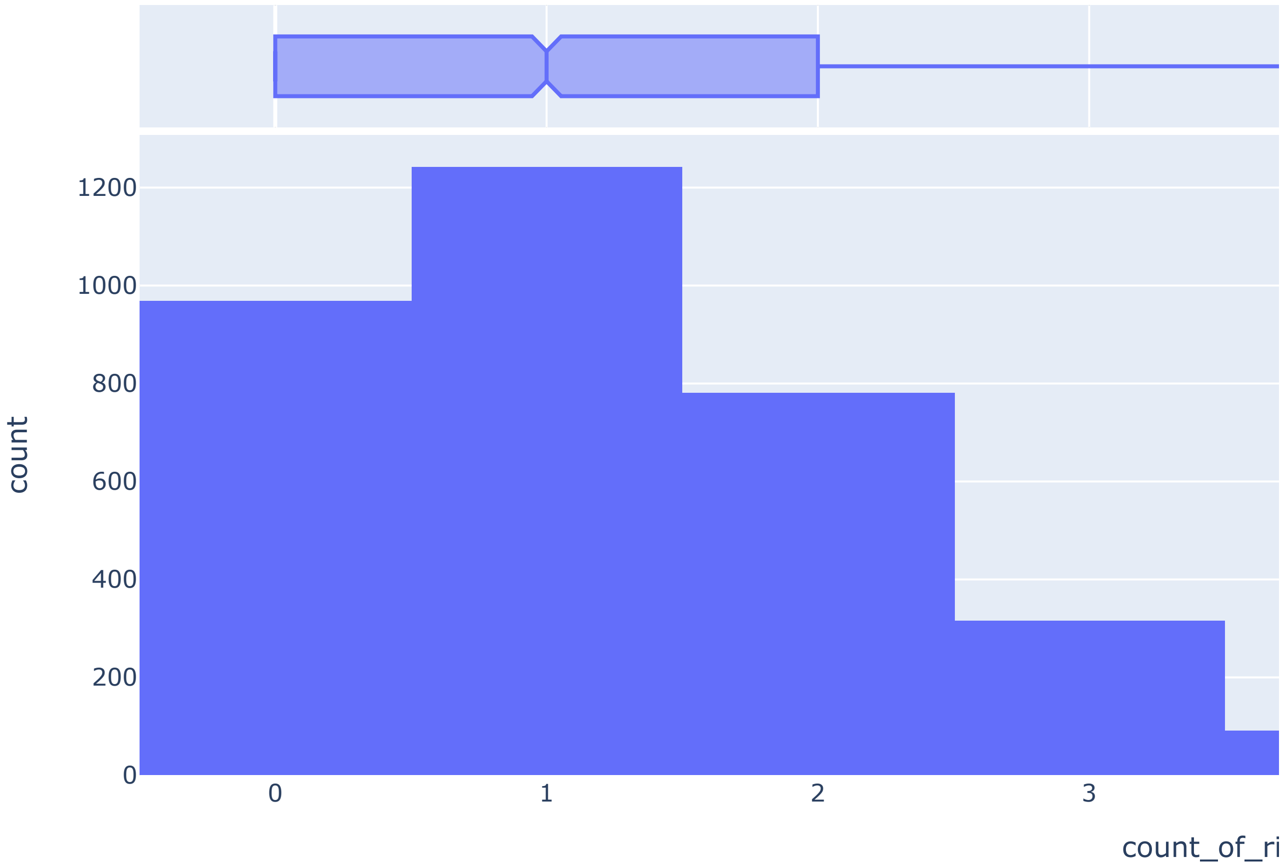
```
gluco_record_during_rx                                                0
dexa_freq_during_rx                                                   0
dexa_during_rx                                                        0
frag_frac_prior_ntm                                                   0
frag_frac_during_rx                                                   0
risk_segment_prior_ntm                                                0
tscore_bucket_prior_ntm                                               0
risk_segment_during_rx                                                0
tscore_bucket_during_rx                                               0
change_t_score                                                        0
change_risk_segment                                                   0
adherent_flag                                                         0
idn_indicator                                                         0
injectable_experience_during_rx                                       0
comorb_encounter_for_screening_for_malignant_neoplasms                0
comorb_encounter_for_immunization                                     0
comorb_encntr_for_general_exam_w_o_complaint,_susp_or_reprtd_dx       0

comorb_vitamin_d_deficiency                                           0
comorb_other_joint_disorder_not_elsewhere_classified                  0
comorb_encntr_for_oth_sp_exam_w_o_complaint_suspected_or_reprtd_dx    0
comorb_long_term_current_drug_therapy                                 0
comorb_dorsalgia                                                      0
comorb_personal_history_of_other_diseases_and_conditions              0
comorb_other_disorders_of_bone_density_and_structure                  0
comorb_disorders_of_lipoprotein_metabolism_and_other_lipidemias       0
comorb_osteoporosis_without_current_pathological_fracture             0
comorb_personal_history_of_malignant_neoplasm                         0
comorb_gastro_esophageal_reflux_disease                               0
```

```
concom_cholesterol_and_triglyceride_regulating_preparations          0
concom_narcotics                                                     0
concom_systemic_corticosteroids_plain                                0
concom_anti_depressants_and_mood_stabilisers                         0
concom_fluoroquinolones                                              0
concom_cephalosporins                                                0
concom_macrolides_and_similar_types                                  0
concom_broad_spectrum_penicillins                                    0
concom_anaesthetics_general                                          0
concom_viral_vaccines                                                0
risk_type_1_insulin_dependent_diabetes                               0
risk_osteogenesis_imperfecta                                         0
risk_rheumatoid_arthritis                                            0
risk_untreated_chronic_hyperthyroidism                               0
risk_untreated_chronic_hypogonadism                                  0
risk_untreated_early_menopause                                       0
risk_patient_parent_fractured_their_hip                              0
risk_smoking_tobacco                                                 0
risk_chronic_malnutrition_or_malabsorption                           0
risk_chronic_liver_disease                                           0
risk_family_history_of_osteoporosis                                  0
risk_low_calcium_intake                                              0
risk_vitamin_d_insufficiency                                         0
risk_poor_health_frailty                                             0
risk_excessive_thinness                                              0
risk_hysterectomy_oophorectomy                                       0
risk_estrogen_deficiency                                             0
risk_immobilization                                                  0
risk_recurring_falls                                                 0
count_of_risks                                                       0
dtype: int64
```

Outlier Analysis

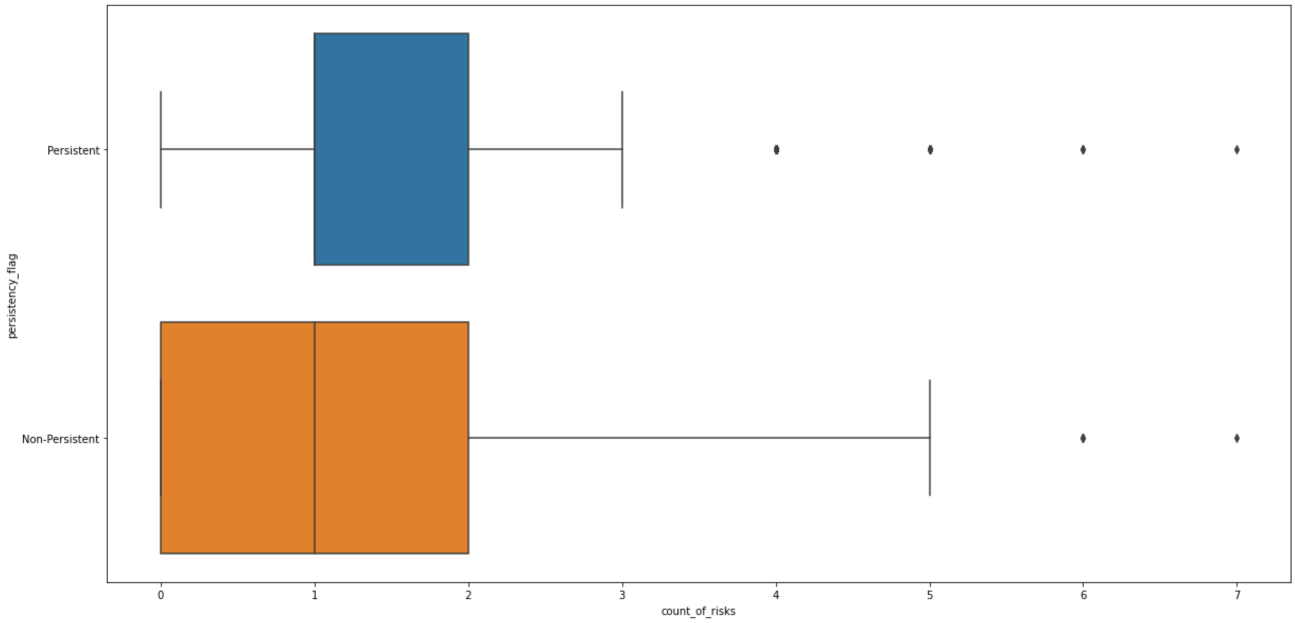
```
fig = px.histogram(df, x="dexa_freq_during_rx",
                   marginal="box", # or violin, rug
                   hover_data=df.columns)
fig.show()
```


```
fig = px.histogram(df, x="count_of_risks",
                   marginal="box", # or violin, rug
                   hover_data=df.columns)
fig.show()
```



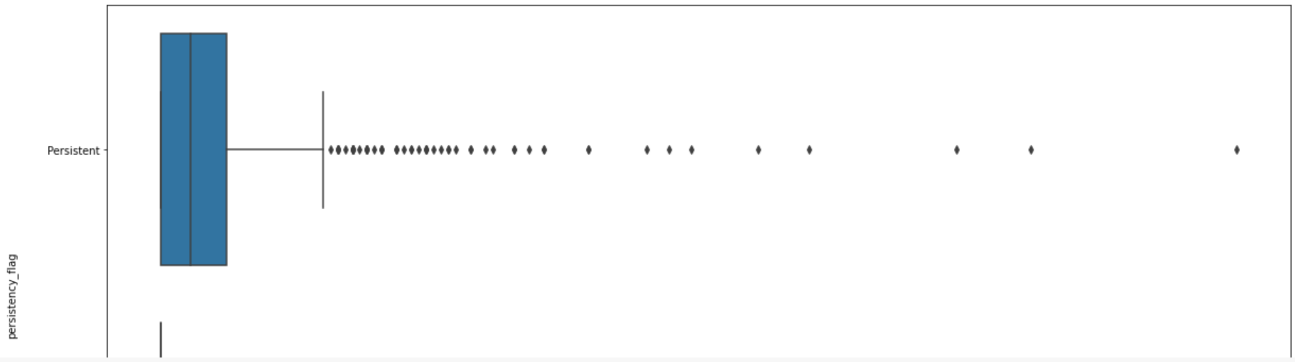

```
plt.figure(figsize=(20,10))
var ="count_of_risks"
sns.boxplot(x=var,y ="persistency_flag",data=df)
```

```
<Figure size 1440x720 with 0 Axes><matplotlib.axes._subplots.AxesSubplot at 0
```




```
plt.figure(figsize=(20,10))
var ="dexa_freq_during_rx"
sns.boxplot(x=var,y ="persistency_flag",data=df)
```

```
<Figure size 1440x720 with 0 Axes><matplotlib.axes._subplots.AxesSubplot at 0
```




```
print("Count of risks skweness: ",df["count_of_risks"].skew())
print("Count of risks Kurtosis: ",df["count_of_risks"].kurt())

## Data shows a moderate positive skewed data on this column and fairly platykurti
## Means the data has little outliers
```

```
Count of risks skweness:  0.8797905232898707
Count of risks Kurtosis:  0.9004859968892842
```

```
print("dexa_freq_during_rx skweness: ",df["dexa_freq_during_rx"].skew())
print("dexa_freq_during_rx Kurtosis: ",df["dexa_freq_during_rx"].kurt())
## very high positive skewed and also with very high kurtosis(Platykurtic)
## This suggests Presence of alot of outliers.
```

```
dexa_freq_during_rx skweness:  6.8087302112992285
dexa_freq_during_rx Kurtosis:  74.75837754795428
```

```
#standardizing dexa_freq_during_rx df
dexa_scaled = StandardScaler().fit_transform(df['dexa_freq_during_rx'][:,np.newaxis
low_range = dexa_scaled[dexa_scaled[:,0].argsort()][:10]
high_range= dexa_scaled[dexa_scaled[:,0].argsort()][-10:]
print('outer range (low) of the distribution:')
print(low_range)
print('\nouter range (high) of the distribution:')
print(high_range)
```

```
outer range (low) of the distribution:
[[-0.3707352]
 [-0.3707352]
 [-0.3707352]
 [-0.3707352]
 [-0.3707352]
 [-0.3707352]
 [-0.3707352]
 [-0.3707352]
 [-0.3707352]
 [-0.3707352]]

outer range (high) of the distribution:
[[ 7.98784109]
 [ 8.11076133]
 [ 8.47952205]
 [ 9.58580421]
 [10.44624589]
 [10.44624589]
```

```
       [12.90465068]
       [13.15049116]
       [14.13385307]
       [17.57561978]]
```

```
scaler = RobustScaler()
df['dexa_freq_during_rx'] = scaler.fit_transform(df['dexa_freq_during_rx'].values.
```

```
scaler = RobustScaler()
df['count_of_risks'] = scaler.fit_transform(df['count_of_risks'].values.reshape(-1
```

```
''' Detection '''
# IQR
Q1 = np.percentile(df['dexa_freq_during_rx'], 25,
                   interpolation = 'midpoint')

Q3 = np.percentile(df['dexa_freq_during_rx'], 75,
                   interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df.shape)

# Upper bound
upper = np.where(df['dexa_freq_during_rx'] >= (Q3+1.5*IQR))
# Lower bound
lower = np.where(df['dexa_freq_during_rx'] <= (Q1-1.5*IQR))

print("lower",lower[0])
print("Upper",upper[0])

''' Removing the Outliers '''
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)

print("New Shape: ", df.shape)

df = df.reset_index(drop=True)
```

```
' Detection '  Old Shape:  (3424, 69)
lower []
Upper [  32   33   62   65   89  101  110  116  164  180  186  194  198  201
  217  241  246  256  264  282  292  303  327  340  349  358  368  369
  373  378  382  390  415  417  426  433  448  457  462  464  480  495
  496  497  505  514  517  541  545  549  563  575  588  589  592  599
  603  605  613  640  646  651  653  656  657  678  684  688  700  705
  710  711  726  728  729  730  759  760  764  765  785  786  804  814
  823  834  847  849  864  870  873  885  909  915  925  926  930  937
  946  978  982  991  994 1006 1008 1016 1042 1061 1073 1074 1076 1113
 1118 1119 1128 1134 1141 1148 1151 1196 1240 1265 1267 1270 1272 1273
 1280 1283 1286 1291 1315 1359 1360 1363 1365 1370 1372 1396 1398 1404
 1448 1474 1513 1524 1533 1539 1546 1550 1554 1555 1564 1566 1570 1576
 1599 1628 1641 1642 1647 1654 1662 1671 1691 1703 1724 1732 1734 1746
 1752 1773 1782 1783 1788 1793 1803 1815 1826 1833 1834 1836 1838 1848
 1852 1854 1870 1876 1895 1901 1904 1909 1910 1914 1915 1919 1920 1928
 1936 1943 1948 1949 1952 1956 1959 1963 1964 1965 1968 1970 1971 1975
 1982 1983 1988 1993 1996 1997 2000 2002 2005 2006 2009 2010 2011 2013
 2015 2016 2020 2024 2028 2029 2030 2031 2033 2034 2038 2041 2042 2043
 2044 2046 2049 2054 2057 2058 2059 2060 2062 2065 2066 2069 2075 2081
```

```
''' Detection '''
# IQR
Q1 = np.percentile(df['count_of_risks'], 25,
                   interpolation = 'midpoint')

Q3 = np.percentile(df['count_of_risks'], 75,
                   interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df.shape)

# Upper bound
upper = np.where(df['count_of_risks'] >= (Q3+1.5*IQR))
# Lower bound
lower = np.where(df['count_of_risks'] <= (Q1-1.5*IQR))

print("lower",lower[0])
print("Upper",upper[0])

''' Removing the Outliers '''
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)

print("New Shape: ", df.shape)

df = df.reset_index(drop=True)
```

```
' Detection '  Old Shape:  (2964, 69)
lower []
Upper [ 281  318  327  507  655  665  678  705  733  952 1001 1126 1590 1624
 1836 2227 2234 2450 2611 2702 2755 2888]
' Removing the Outliers '  New Shape:  (2942, 69)
```

Describe Data

```
#distribution of categorical features
df.describe(include=['O'])
```

	ptid	persistency_flag	gender	race	ethnicity	region	age_bucket
count	2942	2942	2942	2942	2942	2942	2942
unique	2942	2	2	4	3	5	4
top	P857	Non-Persistent	Female	Caucasian	Not Hispanic	Midwest	>75
freq	1	2047	2769	2701	2784	1210	1262

```
df.groupby(['persistency_flag']).mean().T
```

persistency_flag	Non-Persistent	Persistent
dexa_freq_during_rx	0.085491	0.662570
count_of_risks	0.074744	0.155866

```
df.groupby(['gender']).mean().T
```

gender	Female	Male
dexa_freq_during_rx	0.263874	0.215800
count_of_risks	0.099494	0.098266

```
df.groupby(['race']).mean()
```

	dexa_freq_during_rx	count_of_risks
race		
African American	0.246377	0.168478
Asian	0.135266	0.021739
Caucasian	0.266445	0.098297
Other/Unknown	0.204167	0.125000

```
df.groupby(['ethnicity']).mean().T
```

ethnicity	Hispanic	Not Hispanic	Unknown
dexa_freq_during_rx	0.279835	0.260417	0.264069
count_of_risks	0.265432	0.097342	0.000000

```
df.groupby(['age_bucket']).mean().T
```

```
df.groupby(['age_bucket']).mean().T
```

age_bucket	55-65	65-75	<55	>75
dexa_freq_during_rx	0.242229	0.297880	0.273973	0.242208
count_of_risks	0.118167	0.097039	0.089041	0.093106

```
df.groupby(['ntm_speciality']).mean().T
```

ntm_speciality	CARDIOLOGY	CLINICAL NURSE SPECIALIST	EMERGENCY MEDICINE	ENDOCRINOLOGY	GASTROENT
dexa_freq_during_rx	0.285714	0.0	0.0	0.392265	
count_of_risks	0.380952	-0.5	0.0	0.279006	

```
df.groupby(['ntm_specialist_flag']).mean().T
```

ntm_specialist_flag	Others	Specialist
dexa_freq_during_rx	0.215145	0.330765
count_of_risks	0.056370	0.164812

```
df.groupby(['ntm_speciality_bucket']).mean().T
```

ntm_speciality_bucket	Endo/Onc/Uro	OB/GYN/Others/PCP/Unknown	Rheum
dexa_freq_during_rx	0.442907	0.215274	0.221349
count_of_risks	0.170415	0.053639	0.185658

```
df.groupby(['ntm_speciality_bucket']).mean().T
```

ntm_speciality_bucket	Endo/Onc/Uro	OB/GYN/Others/PCP/Unknown	Rheum
dexa_freq_during_rx	0.442907	0.215274	0.221349
count_of_risks	0.170415	0.053639	0.185658

```
df.groupby(['risk_chronic_liver_disease']).mean().T
```

risk_chronic_liver_disease	N	Y
dexa_freq_during_rx	0.260132	0.452381

```
df.groupby(['risk_family_history_of_osteoporosis']).mean().T
```

risk_family_history_of_osteoporosis	N	Y
dexa_freq_during_rx	0.258113	0.287671
count_of_risks	0.045283	0.590753

```
df.groupby(['risk_low_calcium_intake']).mean().T
```

risk_low_calcium_intake	N	Y
dexa_freq_during_rx	0.261069	0.259259
count_of_risks	0.090502	0.819444

```
df.groupby(['risk_vitamin_d_insufficiency']).mean().T
```

risk_vitamin_d_insufficiency	N	Y
dexa_freq_during_rx	0.223363	0.303468
count_of_risks	-0.175866	0.409321

```
df.groupby(['risk_excessive_thinness']).mean().T
```

risk_excessive_thinness	N	Y
dexa_freq_during_rx	0.261946	0.218579
count_of_risks	0.085908	0.737705

```
df.groupby(['risk_hysterectomy_oophorectomy']).mean().T
```

risk_hysterectomy_oophorectomy	N	Y
dexa_freq_during_rx	0.261650	0.222222
count_of_risks	0.089748	0.722222

```
df.groupby(['risk_estrogen_deficiency']).mean().T
```

risk_estrogen_deficiency	N	Y
dexa_freq_during_rx	0.261052	0.259259
count_of_risks	0.097682	0.666667

```
df.groupby(['risk_immobilization']).mean().T
```

risk_immobilization	N	Y
dexa_freq_during_rx	0.262002	0.027778
count_of_risks	0.096416	0.833333

```
df.groupby(['risk_recurring_falls']).mean().T
```

risk_recurring_falls	N	Y
dexa_freq_during_rx	0.259901	0.321212
count_of_risks	0.087634	0.718182

Data Wrangling , Transformation and Standardization

```
df = df.drop(['ptid'], axis=1)
```

```
mapper = {'N': 0, 'Y':1}
df = df.replace(mapper)
```

```
df['persistency_flag'] = df['persistency_flag'].replace(['Non-Persistent', 'Persis
df.head()
```

	persistency_flag	gender	race	ethnicity	region	age_bucket	ntm_s
0	1	Male	Caucasian	Not Hispanic	West	>75	PRA
1	0	Male	Asian	Not Hispanic	West	55-65	PRA
2	0	Female	Other/Unknown	Hispanic	Midwest	65-75	PRA
3	0	Female	Caucasian	Not Hispanic	Midwest	>75	PRA
4	0	Female	Caucasian	Not Hispanic	Midwest	>75	PRA

Analyzing dependency of variable (After Transformation)

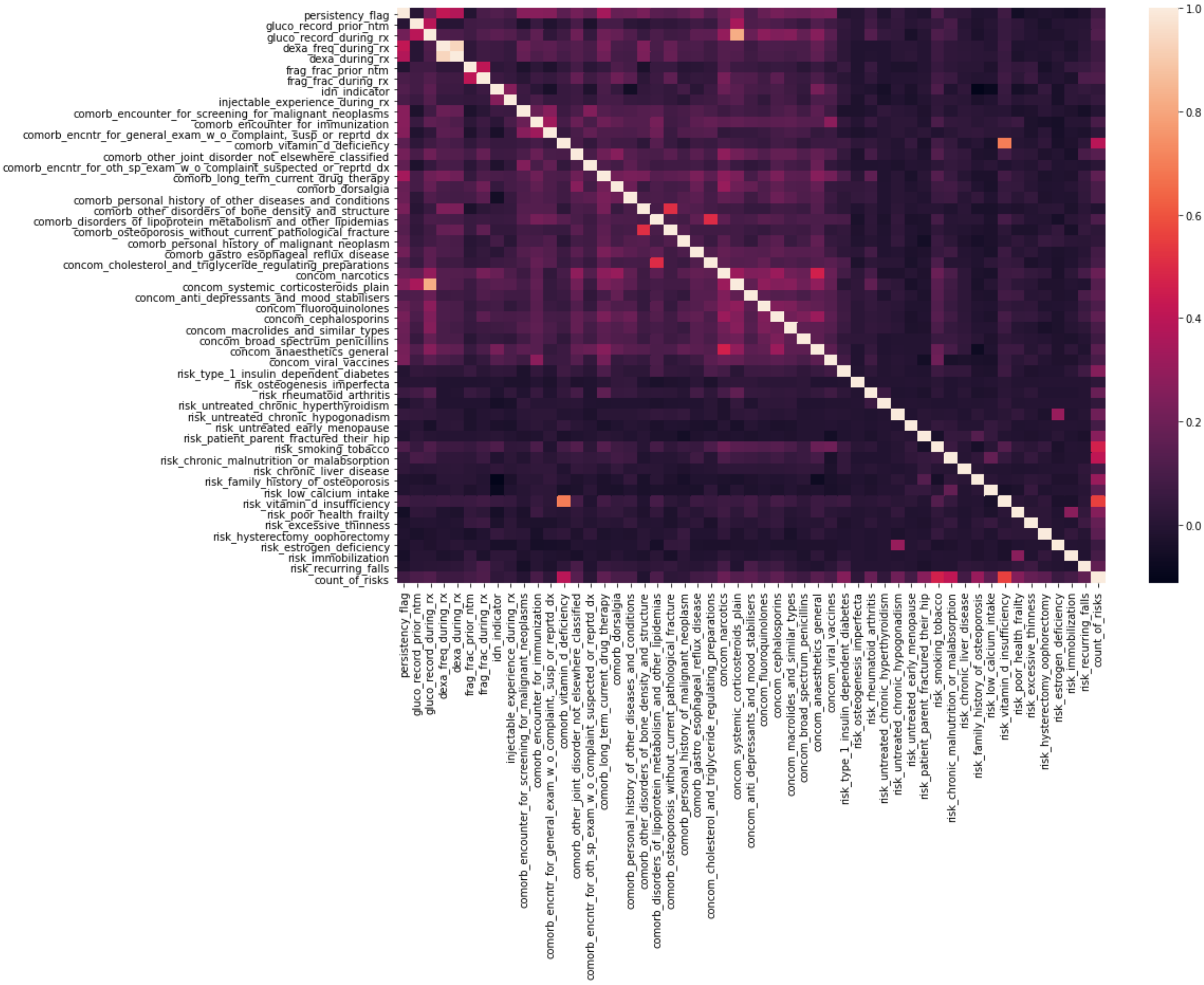
```
np.abs(df.corr()).sort_values(by=['persistency_flag'], ascending=False)
```

	persistency_fla
persistency_flag	1.00000
dexa_freq_during_rx	0.41487
dexa_during_rx	0.37496
comorb_long_term_current_drug_therapy	0.34277
comorb_encounter_for_screening_for_malignant_neoplasms	0.26833
comorb_encounter_for_immunization	0.26830
comorb_encntr_for_general_exam_w_o_complaint,_susp_or_reprtd_dx	0.25789
concom_systemic_corticosteroids_plain	0.24904
concom_viral_vaccines	0.22700
comorb_other_disorders_of_bone_density_and_structure	0.22700
concom_anaesthetics_general	0.22061
concom_cephalosporins	0.21782
comorb_other_joint_disorder_not_elsewhere_classified	0.21593
gluco_record_during_rx	0.21277
comorb_gastro_esophageal_reflux_disease	0.20798
concom_macrolides_and_similar_types	0.19235
comorb_personal_history_of_other_diseases_and_conditions	0.18956
concom_narcotics	0.18839
concom_broad_spectrum_penicillins	0.18661
concom_fluoroquinolones	0.18121
comorb_dorsalgia	0.17992
comorb_encntr_for_oth_sp_exam_w_o_complaint_suspected_or_reprtd_dx	0.16409
comorb_personal_history_of_malignant_neoplasm	0.15727
comorb_vitamin_d_deficiency	0.15159
comorb_disorders_of_lipoprotein_metabolism_and_other_lipidemias	0.1474
comorb_osteoporosis_without_current_pathological_fracture	0.13264
idn_indicator	0.12588
concom_cholesterol_and_triglyceride_regulating_preparations	0.12532
risk_smoking_tobacco	0.11557
concom_anti_depressants_and_mood_stabilisers	0.11172
frag_frac_during_rx	0.10294
injectable_experience_during_rx	0.09749
count_of_risks	0.0715

count_of_risks	0.0715
risk_vitamin_d_insufficiency	0.0695
risk_rheumatoid_arthritis	0.0595
risk_poor_health_frailty	0.0558
risk_untreated_chronic_hypogonadism	0.0452
risk_immobilization	0.0423
risk_chronic_malnutrition_or_malabsorption	0.0316
risk_chronic_liver_disease	0.0294
risk_excessive_thinness	0.0236
risk_estrogen_deficiency	0.0232
risk_recurring_falls	0.0203
risk_untreated_chronic_hyperthyroidism	0.0172
risk_family_history_of_osteoporosis	0.0168
risk_hysterectomy_oophorectomy	0.0161
risk_patient_parent_fractured_their_hip	0.0150
risk_low_calcium_intake	0.0131
risk_type_1_insulin_dependent_diabetes	0.0071
frag_frac_prior_ntm	0.0055
risk_untreated_early_menopause	0.0041
gluco_record_prior_ntm	0.0030
risk_osteogenesis_imperfecta	0.0020

```
plt.subplots(figsize=(15,10))
sns.heatmap(df.corr())
```

```
(<Figure size 1080x720 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7fc0115e5390>)<matplotlib.axes._
```



Creating Dummy values

```
X=df.drop(['persistency_flag'],axis=1)
y=df['persistency_flag']

X = pd.get_dummies(X)
X.columns=[x.lower() for x in X.columns]
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=42,test_size=0.3, 
```

```
df_train = X_train.copy()
df_train['persistency_flag'] = y_train
df_train.head()
```

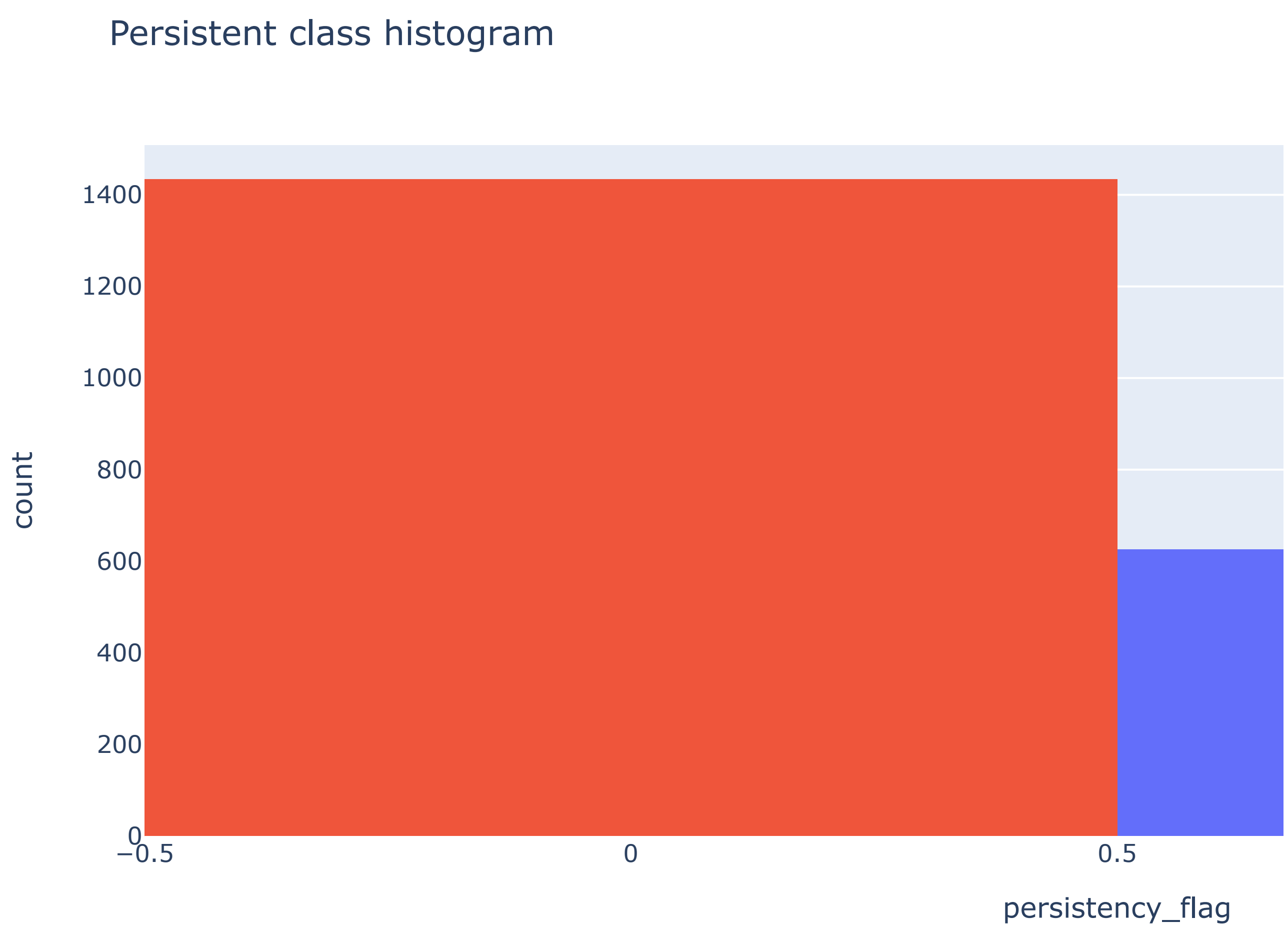
	gluco_record_prior_ntm	gluco_record_during_rx	dexa_freq_during_rx	dex
1493	1	1	0.0	
1375	0	0	0.0	
1217	1	1	0.0	
1157	1	1	0.0	

Come Imbalanced dataset

```
classes=df_train['persistency_flag'].value_counts()
normal_share=round(classes[0]/df_train['persistency_flag'].count()*100,2)
fraud_share=round(classes[1]/df_train['persistency_flag'].count()*100, 2)
print("Non-Persistent : {} %".format(normal_share))
print("Persistent : {} %".format(fraud_share))
```

```
Non-Persistent : 69.6 %
Persistent : 30.4 %
```

```
fig = px.histogram(df_train, x="persistency_flag", color="persistency_flag", title=
fig.show()
```



Upsampling

```
# Upsampling
df_minority_upsampled = resample(df_train[df_train['persistency_flag'] == 1],
                                 replace=True,     # sample with replacement
                                 n_samples=len(df_train[df_train['persistency_flag
                                 random_state=123) # reproducible results

# Combine majority class with upsampled minority class
df_train = pd.concat([df_train[df_train['persistency_flag'] == 0], df_minority_ups

# Display new class counts
df_train.persistency_flag.value_counts()
```

```
1    1433
0    1433
Name: persistency_flag, dtype: int64
```

```
X_train=df_train.drop(['persistency_flag'],axis=1)
y_train=df_train['persistency_flag']
```

```
fig = px.histogram(df_train, x="persistency_flag", color="persistency_flag", title=
fig.show()
```

Model Creation

Linear Models

LogisticRegression

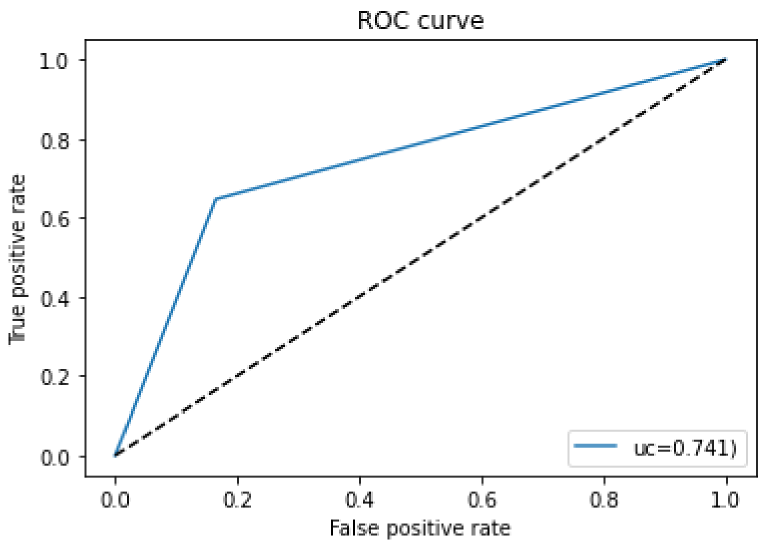
```
logistic(X_train,X_test,y_train,y_test)
```

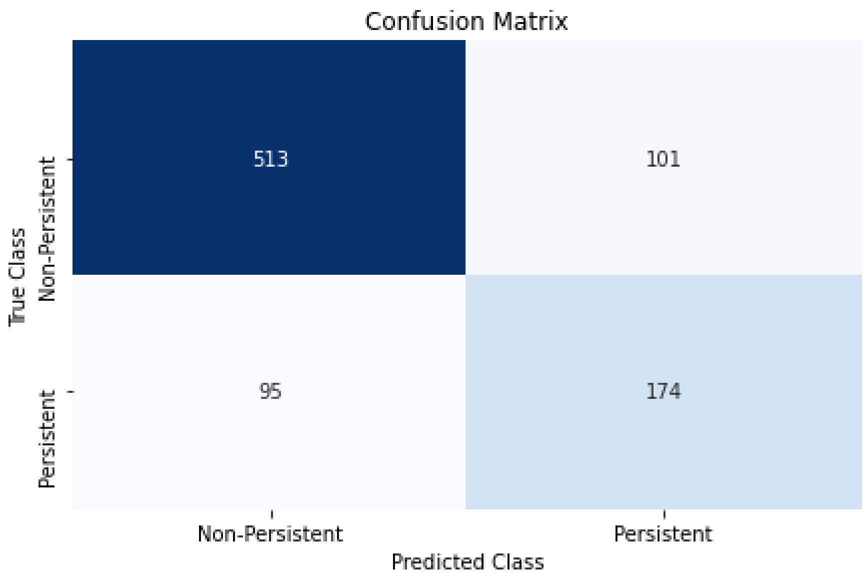
```
Accuracy : 0.7814269535673839
Precision : 0.6283783783783784
Recall : 0.6914498141263941
F1 Score : 0.6584070796460177
```

RidgeClassifier

```
Ridge(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.7780294450736127
Precision : 0.6327272727272727
Recall : 0.6468401486988847
F1 Score : 0.6397058823529411
                precision    recall  f1-score   support

Non-Persistent       0.84      0.84      0.84       614
    Persistent       0.63      0.65      0.64       269

      accuracy                           0.78       883
     macro avg       0.74      0.74      0.74       883
  weighted avg       0.78      0.78      0.78       883

AUC : 0.7411725173461852
```

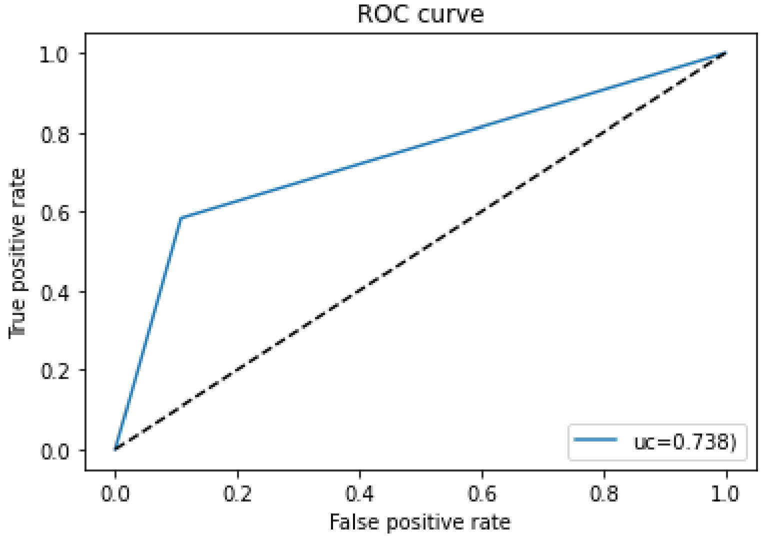


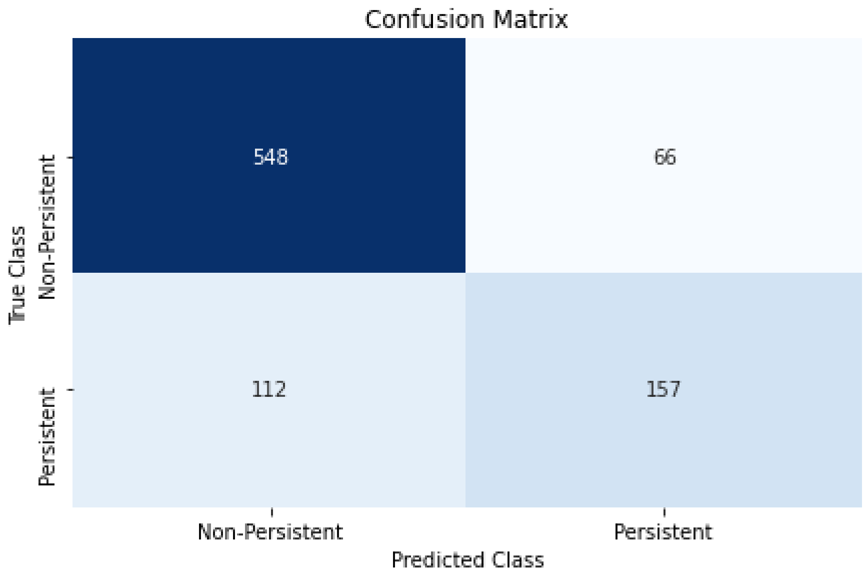


SGDClassifier

```
SGD(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.79841449603624
Precision : 0.7040358744394619
Recall : 0.5836431226765799
F1 Score : 0.6382113821138212
                precision    recall  f1-score   support

Non-Persistent       0.83      0.89      0.86       614
    Persistent       0.70      0.58      0.64       269

      accuracy                           0.80       883
     macro avg       0.77      0.74      0.75       883
  weighted avg       0.79      0.80      0.79       883

AUC : 0.7380756329995277
```



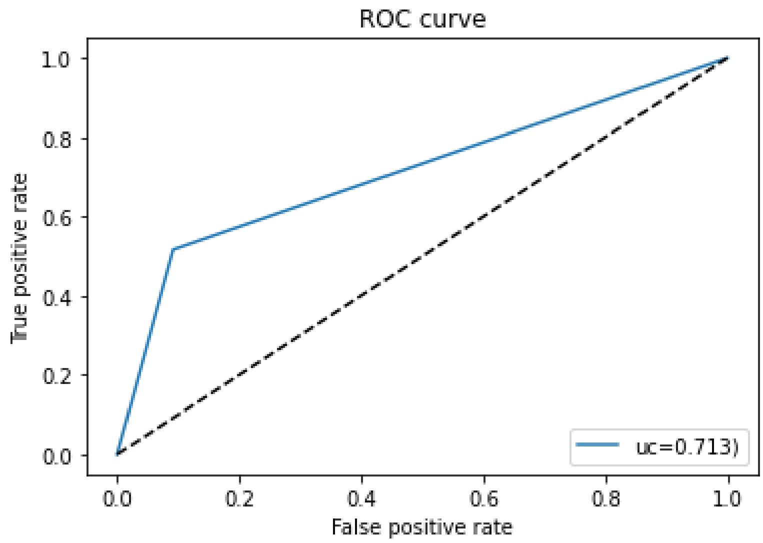


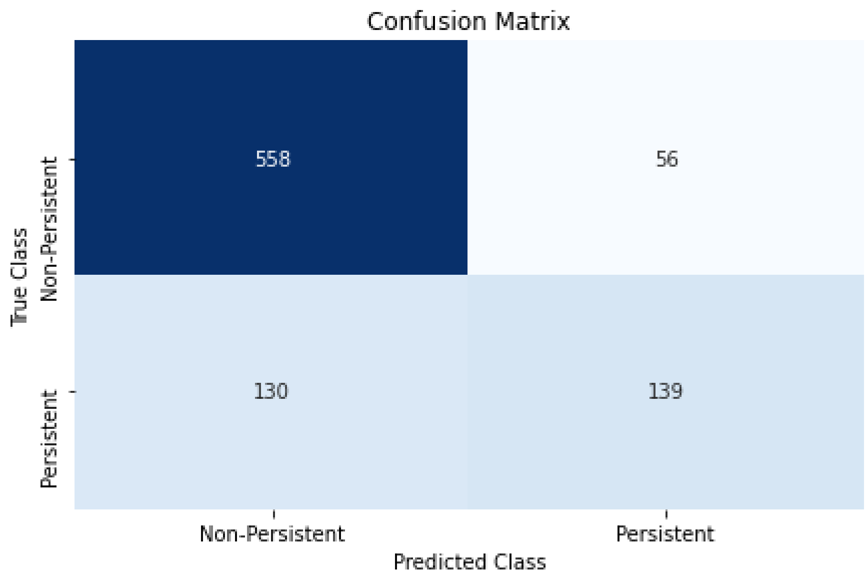
Ensemble and Boosting Models

RandomForestClassifier

```
RF(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.7893544733861835
Precision : 0.7128205128205128
Recall : 0.516728624535316
F1 Score : 0.5991379310344827
                precision    recall  f1-score   support

Non-Persistent       0.81      0.91      0.86       614
    Persistent       0.71      0.52      0.60       269

      accuracy                           0.79       883
     macro avg       0.76      0.71      0.73       883
  weighted avg       0.78      0.79      0.78       883

AUC : 0.7127617064044658
```

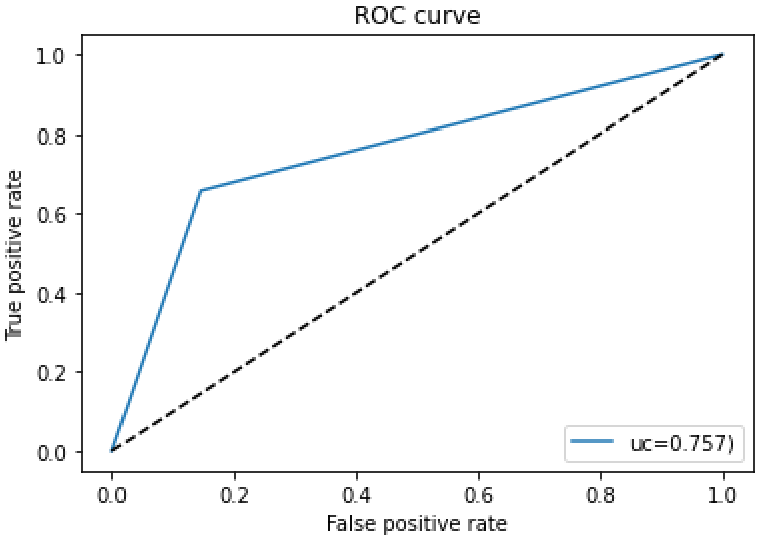


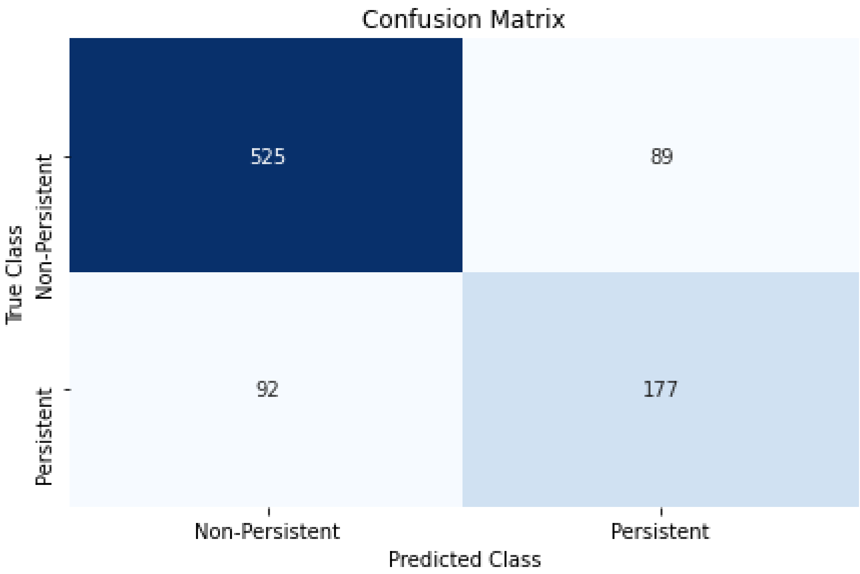


BaggingClassifier

```
Bagging(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.7950169875424689
Precision : 0.6654135338345865
Recall : 0.6579925650557621
F1 Score : 0.6616822429906541
                precision    recall  f1-score   support

Non-Persistent       0.85      0.86      0.85       614
    Persistent       0.67      0.66      0.66       269

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

AUC : 0.7565207124953077
```

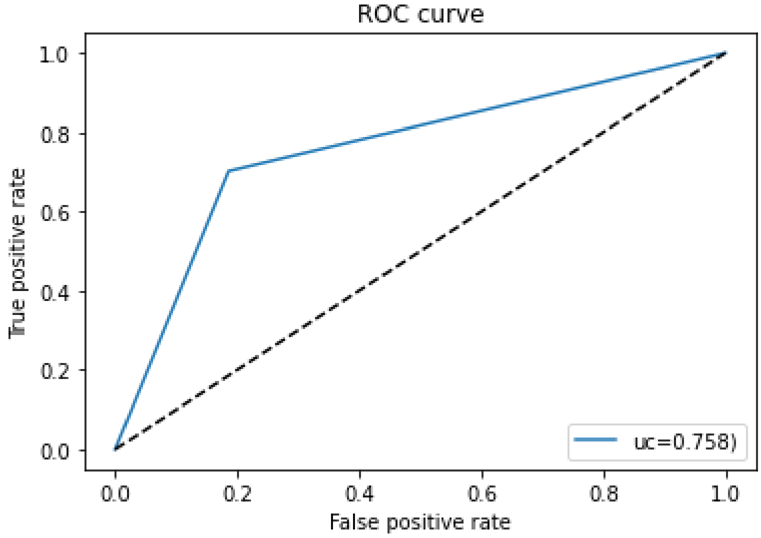


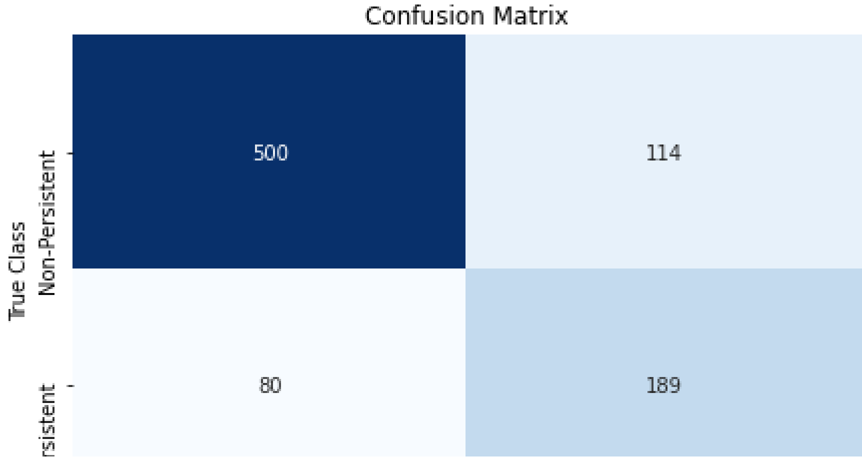


AdaBoostClassifier

```
AdaBoost(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.7802944507361268
Precision : 0.6237623762376238
Recall : 0.7026022304832714
F1 Score : 0.6608391608391608
                precision    recall  f1-score   support

Non-Persistent       0.86      0.81      0.84       614
    Persistent       0.62      0.70      0.66       269

      accuracy                           0.78       883
     macro avg       0.74      0.76      0.75       883
  weighted avg       0.79      0.78      0.78       883

AUC : 0.7584672390201372
```

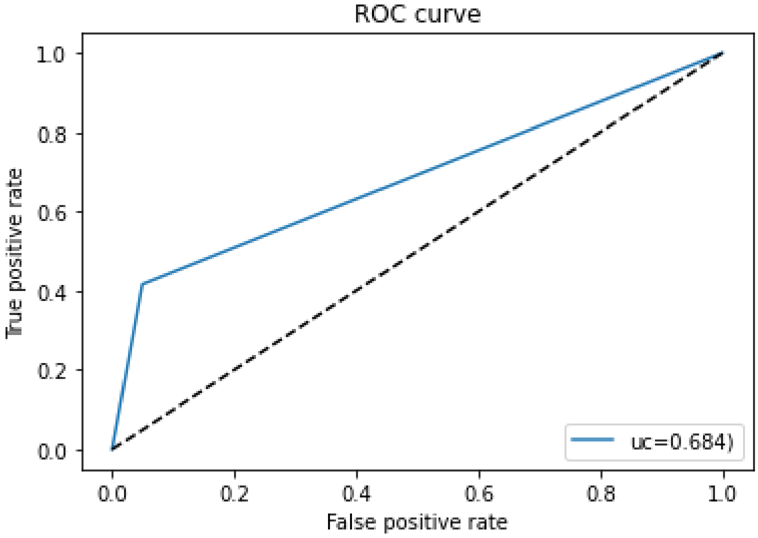


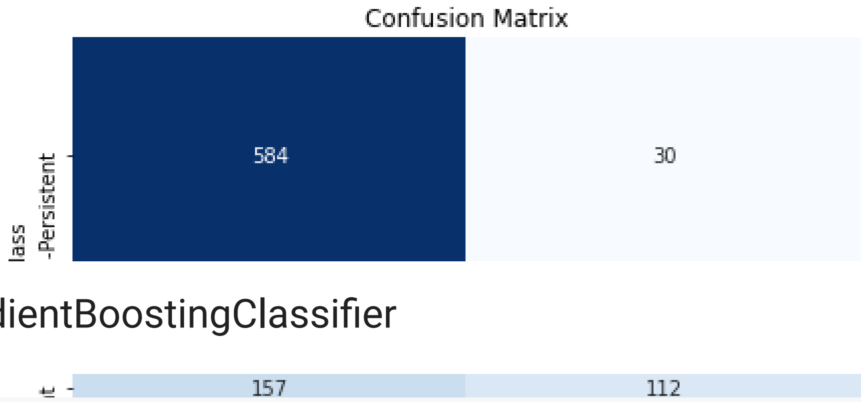


ExtraTreesClassifier

```
ExtraTrees(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.7882219705549264
Precision : 0.7887323943661971
Recall : 0.4163568773234201
F1 Score : 0.5450121654501217
                precision    recall  f1-score   support

Non-Persistent       0.79      0.95      0.86       614
    Persistent       0.79      0.42      0.55       269

      accuracy                           0.79       883
     macro avg       0.79      0.68      0.70       883
  weighted avg       0.79      0.79      0.77       883

AUC : 0.6837484712349999
```

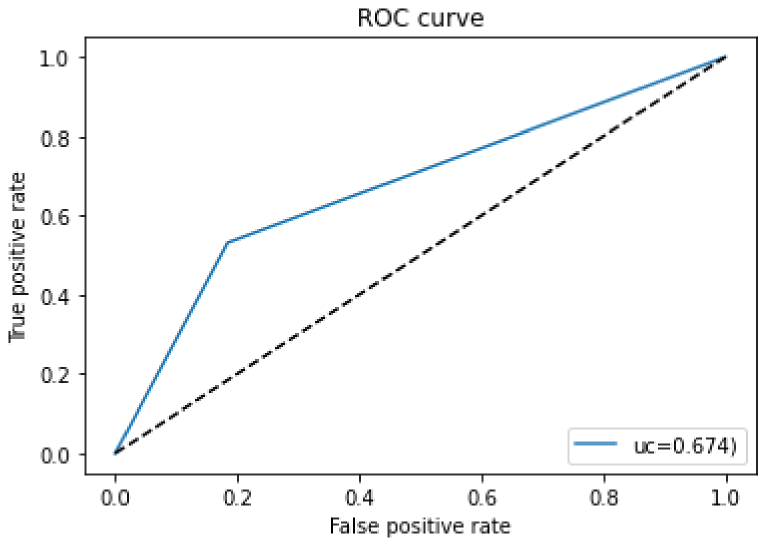




GradientBoostingClassifier

```
GradientBoosting(X_train,X_test,y_train,y_test)
```

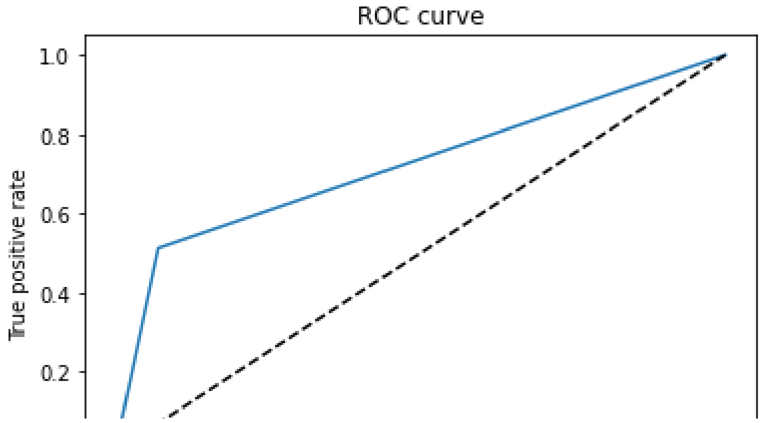
```
Accuracy : 0.7293318233295584
Precision : 0.55859375
Recall : 0.5315985130111525
F1 Score : 0.5447619047619048
                precision    recall  f1-score   support

Non-Persistent       0.80      0.82      0.81       614
    Persistent       0.56      0.53      0.54       269

      accuracy                           0.73       883
     macro avg       0.68      0.67      0.68       883
  weighted avg       0.73      0.73      0.73       883

AUC : 0.6737797125316348
```



StackingClassifier

```
Stacking(X_train,X_test,y_train,y_test)
```

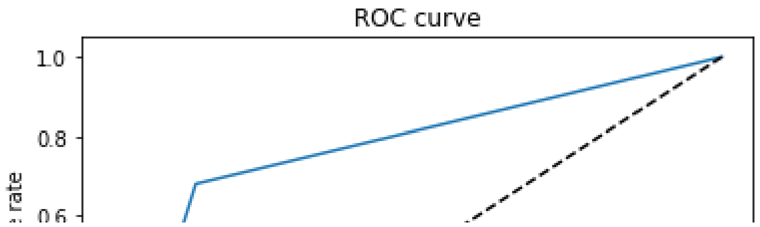
```
Accuracy : 0.8029445073612684
Precision : 0.7624309392265194
Recall : 0.5130111524163569
F1 Score : 0.6133333333333333
                precision    recall  f1-score   support

Non-Persistent       0.81      0.93      0.87       614
    Persistent       0.76      0.51      0.61       269

      accuracy                           0.80       883
     macro avg       0.79      0.72      0.74       883
  weighted avg       0.80      0.80      0.79       883

AUC : 0.7214892895632273
```



XGBoostClassifier

```
XGBOOST(X_train.values,X_test.values,y_train,y_test)
```

```
Accuracy : 0.8074745186862967
Precision : 0.6853932584269663
Recall : 0.6802973977695167
F1 Score : 0.6828358208955224
                precision    recall  f1-score   support

Non-Persistent       0.86      0.86      0.86       614
    Persistent       0.69      0.68      0.68       269

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

AUC : 0.7717447900899701
```



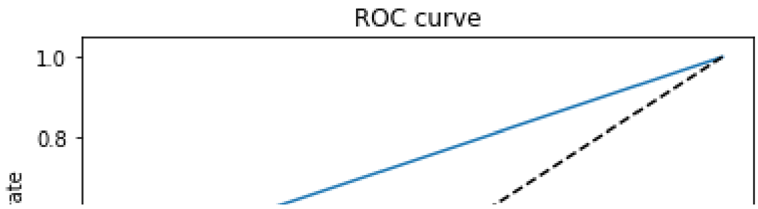
Neural Network

Multi Layer Perceptron

False positive rate

```
MLP(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.7587768969422424
Precision : 0.6129032258064516
Recall : 0.5650557620817844
F1 Score : 0.5880077369439072
                precision    recall  f1-score   support

Non-Persistent       0.82      0.84      0.83       614
    Persistent       0.61      0.57      0.59       269

      accuracy                           0.76       883
     macro avg       0.71      0.70      0.71       883
  weighted avg       0.75      0.76      0.76       883

AUC : 0.70435198527542
```



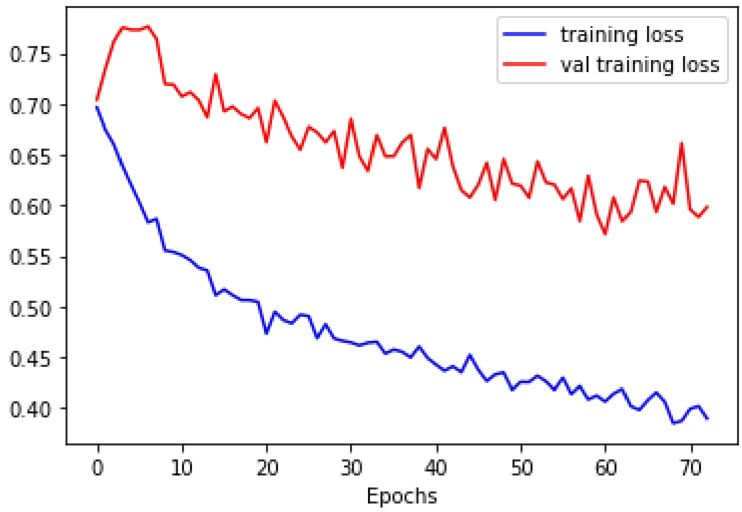
Multilayer Neural Network with Tensorflow/Keras

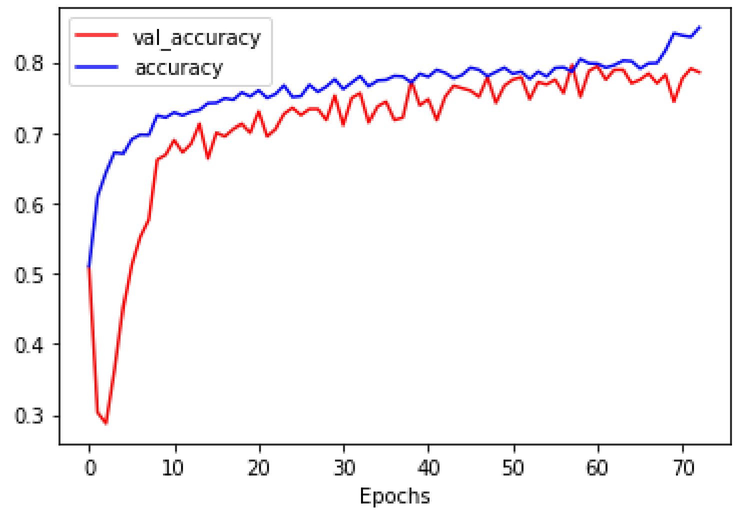
```
MNN(X_train,X_test,y_train,y_test)
```

```
Epoch 1/150
230/230 [==============================] - 16s 4ms/step - loss: 0.6977 - accu
Epoch 2/150
230/230 [==============================] - 0s 2ms/step - loss: 0.6781 - accur
Epoch 3/150
230/230 [==============================] - 0s 2ms/step - loss: 0.6631 - accur
Epoch 4/150
230/230 [==============================] - 0s 1ms/step - loss: 0.6396 - accur
Epoch 5/150
230/230 [==============================] - 0s 1ms/step - loss: 0.6184 - accur
Epoch 6/150
230/230 [==============================] - 0s 1ms/step - loss: 0.6146 - accur
Epoch 7/150
230/230 [==============================] - 0s 2ms/step - loss: 0.5837 - accur
Epoch 8/150
230/230 [==============================] - 0s 2ms/step - loss: 0.6070 - accur
Epoch 9/150
230/230 [==============================] - 0s 2ms/step - loss: 0.5473 - accur
Epoch 10/150
230/230 [==============================] - 0s 1ms/step - loss: 0.5634 - accur
Epoch 11/150
230/230 [==============================] - 0s 2ms/step - loss: 0.5437 - accur
Epoch 12/150
230/230 [==============================] - 0s 1ms/step - loss: 0.5418 - accur
Epoch 13/150
230/230 [==============================] - 0s 2ms/step - loss: 0.5419 - accur
Epoch 14/150
230/230 [==============================] - 0s 2ms/step - loss: 0.5478 - accur
Epoch 15/150
230/230 [==============================] - 0s 2ms/step - loss: 0.5053 - accur
Epoch 16/150
230/230 [==============================] - 0s 1ms/step - loss: 0.5338 - accur
Epoch 17/150
230/230 [==============================] - 0s 1ms/step - loss: 0.5260 - accur
Epoch 18/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4981 - accur
Epoch 19/150
230/230 [==============================] - 0s 2ms/step - loss: 0.5274 - accur
Epoch 20/150
230/230 [==============================] - 0s 1ms/step - loss: 0.5177 - accur
Epoch 21/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4841 - accur
Epoch 22/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4774 - accur
Epoch 23/150
230/230 [==============================] - 0s 2ms/step - loss: 0.5042 - accur
Epoch 24/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4919 - accur
Epoch 25/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4803 - accur
Epoch 26/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4945 - accur
Epoch 27/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4544 - accur
Epoch 28/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4854 - accur
Epoch 29/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4741 - accur
Epoch 30/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4663 - accur
Epoch 31/150
```

```
230/230 [==============================] - 0s 1ms/step - loss: 0.4556 - accur
Epoch 32/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4630 - accur
Epoch 33/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4495 - accur
Epoch 34/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4712 - accur
Epoch 35/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4543 - accur
Epoch 36/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4587 - accur
Epoch 37/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4532 - accur
Epoch 38/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4557 - accur
Epoch 39/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4462 - accur
Epoch 40/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4588 - accur
Epoch 41/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4308 - accur
Epoch 42/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4271 - accur
Epoch 43/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4316 - accur
Epoch 44/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4246 - accur
Epoch 45/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4610 - accur
Epoch 46/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4263 - accur
Epoch 47/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4398 - accur
Epoch 48/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4304 - accur
Epoch 49/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4523 - accur
Epoch 50/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4195 - accur
Epoch 51/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4190 - accur
Epoch 52/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4166 - accur
Epoch 53/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4190 - accur
Epoch 54/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4215 - accur
Epoch 55/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4269 - accur
Epoch 56/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4223 - accur
Epoch 57/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4220 - accur
Epoch 58/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4189 - accur
Epoch 59/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4022 - accur
Epoch 60/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4053 - accur
Epoch 61/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4124 - accur
Epoch 62/150
```

```
Epoch 62/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4249 - accur
Epoch 63/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4215 - accur
Epoch 64/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4038 - accur
Epoch 65/150
230/230 [==============================] - 0s 1ms/step - loss: 0.3922 - accur
Epoch 66/150
230/230 [==============================] - 0s 2ms/step - loss: 0.4065 - accur
Epoch 67/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4115 - accur
Epoch 68/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4012 - accur
Epoch 69/150
230/230 [==============================] - 0s 1ms/step - loss: 0.3669 - accur
Epoch 70/150
230/230 [==============================] - 0s 1ms/step - loss: 0.3899 - accur
Epoch 71/150
230/230 [==============================] - 0s 1ms/step - loss: 0.4225 - accur
Epoch 72/150
230/230 [==============================] - 0s 1ms/step - loss: 0.3911 - accur
Epoch 73/150
230/230 [==============================] - 0s 1ms/step - loss: 0.3795 - accur
Epoch 00073: early stopping
```






```
Accuracy : 0.8029445073612684
Precision : 0.6923076923076923
Recall : 0.6356877323420075
F1 Score : 0.6627906976744186
                     precision    recall  f1-score   support

Non-Persistent       0.85      0.88      0.86       614
```

```
Non-Persistent       0.85      0.88      0.86       614
    Persistent       0.69      0.64      0.66       269

      accuracy                           0.80       883
     macro avg       0.77      0.76      0.76       883
  weighted avg       0.80      0.80      0.80       883

AUC : 0.7559546153566713
```

ROC curve

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

Training the final model

```
###Stacking classifier
import pickle
estimators = [('rf', RandomForestClassifier(n_estimators=10, random_state=42)), ('
final_model = StackingClassifier(estimators=estimators, final_estimator=LogisticRe
final_model.fit(X, y)
filename = 'final_model.sav'
pickle.dump(final_model, open(filename, 'wb'))
```

```
StackingClassifier(cv=None,
                   estimators=[('rf',
                                RandomForestClassifier(bootstrap=True,
                                                       ccp_alpha=0.0,
                                                       class_weight=None,
                                                       criterion='gini',
                                                       max_depth=None,
                                                       max_features='auto',
                                                       max_leaf_nodes=None,
                                                       max_samples=None,
                                                       min_impurity_decrease=
                                                       min_impurity_split=Non
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_le
                                                       n_estimators=10,
                                                       n_jobs=None,...
                                                           tol=0.0001,
                                                           verbose=0))],
                                          verbose=False))],
                   final_estimator=LogisticRegression(C=1.0, class_weight=Non
```

```
                                                       dual=False,
                                                       fit_intercept=True,
                                                       intercept_scaling=1,
                                                       l1_ratio=None,
                                                       max_iter=100,
                                                       multi_class='auto',
                                                       n_jobs=None, penalty='l
                                                       random_state=None,
                                                       solver='lbfgs',
                                                       tol=0.0001, verbose=0,
                                                       warm_start=False),
                   n_jobs=None, passthrough=False, stack_method='auto',
                   verbose=0)
```

✓ 5s completed at 3:50 PM