Week 13 Project Deliverable

Group Name: Mike
Member Name: Mike Wang
Email: yuqiao.wang@vanderbilt.edu
Country: USA
College: Vanderbilt University
Batch Code: LISUM21
Specialization: Data Science
Date: 7/29/2023
Submitted to: Data Glacier

Problem Statement

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

The main objective is to build a classification model to predict the NTM drug persistence of patients based on several factors.

Project Schedule

Week	**Date**	**Goal**
07	06/19/2023	Problem Statement, Data Preview
08	06/26/2023	Data Preprocessing
09	07/02/2023	Data Prepare and cleaning
10	07/09/2023	EDA
11	07/16/2023	Recommendation and Model Suggestion
12	07/23/2023	Model Building and Evaluation

13	07/30/2023	Presentation

Data Understanding

Predictor Variables

Demographic features

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

Patient features

IDN Indicator	Flag indicating patients mapped to IDN
NTM - Physician Specialty	Specialty of the HCP that prescribed the NTM Rx
NTM - T-Score	T Score of the patient at the time of the NTM ...
Change in T Score	Change in Tscore before starting with any ther...
NTM - Risk Segment	Risk Segment of the patient at the time of the...
Change in Risk Segment	Change in Risk Segment before starting with an...
NTM - Multiple Risk Factors	Flag indicating if patient falls under multip...
NTM - Dexa Scan Frequency	Number of DEXA scans taken prior to the first ...
NTM - Dexa Scan Recency	Flag indicating the presence of Dexa Scan befo...
Dexa During Therapy	Flag indicating if the patient had a Dexa Scan...
NTM - Fragility Fracture Recency	Flag indicating if the patient had a recent fr...
Fragility Fracture During Therapy	Flag indicating if the patient had fragility f...
NTM - Glucocorticoid Recency	Flag indicating usage of Glucocorticoids (>=7....
Glucocorticoid Usage During Therapy	Flag indicating if the patient had a Glucocort...
NTM - Injectable Experience	Flag indicating any injectable drug usage in t...
NTM - Risk Factors	Risk Factors that the patient is falling into....
NTM - Comorbidity	Comorbidities are divided into two main catego...
NTM - Concomitancy	Concomitant drugs recorded prior to starting w...
Adherence	Adherence for the therapies

Target Variable:

Persistency_Flag	Flag indicating if a patient was persistent or...

Boolean Predictor Variable List: (All the variables in the list have two values of Yes or No)

```
['Gluco_Record_Prior_Ntm',
 'Gluco_Record_During_Rx',
 'Dexa_During_Rx',
 'Frag_Frac_Prior_Ntm',
 'Frag_Frac_During_Rx',
 'Idn_Indicator',
 'Injectable_Experience_During_Rx',
 'Comorb_Encounter_For_Screening_For_Malignant_Neoplasms',
 'Comorb_Encounter_For_Immunization',
 'Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx',
 'Comorb_Vitamin_D_Deficiency',
 'Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified',
 'Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx',
 'Comorb_Long_Term_Current_Drug_Therapy',
 'Comorb_Dorsalgia',
 'Comorb_Personal_History_Of_Other_Diseases_And_Conditions',
 'Comorb_Other_Disorders_Of_Bone_Density_And_Structure',
 'Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias',
 'Comorb_Osteoporosis_without_current_pathological_fracture',
 'Comorb_Personal_history_of_malignant_neoplasm',
 'Comorb_Gastro_esophageal_reflux_disease',
 'Concom_Cholesterol_And_Triglyceride_Regulating_Preparations',
 'Concom_Narcotics',
 'Concom_Systemic_Corticosteroids_Plain',
 'Concom_Anti_Depressants_And_Mood_Stabilisers',
 'Concom_Fluoroquinolones',
 'Concom_Cephalosporins',
 'Concom_Macrolides_And_Similar_Types',
 'Concom_Broad_Spectrum_Penicillins',
 'Concom_Anaesthetics_General',
 'Concom_Viral_Vaccines',
 'Risk_Type_1_Insulin_Dependent_Diabetes',
 'Risk_Osteogenesis_Imperfecta',
 'Risk_Rheumatoid_Arthritis',
 'Risk_Untreated_Chronic_Hyperthyroidism',
 'Risk_Untreated_Chronic_Hypogonadism',
 'Risk_Untreated_Early_Menopause',
 'Risk_Patient_Parent_Fractured_Their_Hip',
 'Risk_Smoking_Tobacco',
 'Risk_Chronic_Malnutrition_Or_Malabsorption',
 'Risk_Chronic_Liver_Disease',
 'Risk_Family_History_Of_Osteoporosis',
 'Risk_Low_Calcium_Intake',
 'Risk_Vitamin_D_Insufficiency',
 'Risk_Poor_Health_Frailty',
 'Risk_Excessive_Thinness',
 'Risk_Hysterectomy_Oophorectomy',
 'Risk_Estrogen_Deficiency',
 'Risk_Immobilization',
 'Risk_Recurring_Falls']
```

All these data will be dummy coded with 0 and 1 values for future classification model

Columns with quantitative data:

	Dexa_Freq_During_Rx	Count_Of_Risks
0	0	0
1	0	0
2	0	2
3	0	1
4	0	1
...	...	...
3419	0	1
3420	0	0
3421	7	1
3422	0	0
3423	0	1

	Dexa_Freq_During_Rx	Count_Of_Risks
count	3424.000000	3424.000000
mean	3.016063	1.239486
std	8.136545	1.094914
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	1.000000
75%	3.000000	2.000000
max	146.000000	7.000000

No outliers and missing values exist in the dataset

Data Cleaning and Transformation

Since there is no missing data in the dataset, the only values that need to be cleaned was features that contains ‘Unknown’ Values:

```
# impute unknown value
df.loc[df['Change_Risk_Segment'] == 'Unknown', 'Change_Risk_Segment'] = 'No change'
df.loc[df['Change_T_Score'] == 'Unknown', 'Change_T_Score'] = 'No change'
```

For unknown changes, I grouped them all into unchanged group which is the majority of the variable

```
def transform_speciality(value):
#     transform medical speciality
    if 'MEDICINE' in value.split(' '):
        return 'MEDICINE'
    elif 'SURGERY' in value.split(' '):
        return 'SURGERY'
    elif df['Ntm_Speciality'].value_counts()[value] < 10 or value == 'Unknown':
        return 'OTHER'
    return value
```

Cleaned and transform medical specialty to decrease the variable numbers.

Use label encoding and one hot encoding to transform different types of categorical features

```
# label encoding variables with only two values to 0 and 1
label(df, 'Gender')
label(df, 'Ntm_Specialist_Flag')
label(df, 'Risk_Segment_Prior_Ntm')
label(df, 'Tscore_Bucket_Prior_Ntm')
label(df, 'Adherent_Flag')
```

```
ohe = onehot_encoder.fit(df[['Ethnicity','Age_Bucket','Ntm_Speciality_Bucket',
                                                      'Race','Risk_Segment_During_Rx','Tscore_Bucket_During_Rx',
                                                       'Change_T_Score','Change_Risk_Segment','Ntm_Speciality']])
```

```
# one hot encode other features into 0 or 1

new_df = pd.DataFrame(ohe.transform(df[['Ethnicity','Age_Bucket','Ntm_Speciality_Bucket',
                                                      'Race','Risk_Segment_During_Rx','Tscore_Bucket_During_Rx',
                                                       'Change_T_Score','Change_Risk_Segment', 'Ntm_Speciality']] ).toarray(),
                      columns=ohe.get_feature_names_out())

new_df.head()
```

	Ethnicity_Hispanic	Ethnicity_Not Hispanic	Ethnicity_Unknown	Age_Bucket_55-65	Age_Bucket_65-75	Age_Bucket_<55	Age_Bucket_>75	Ntm_Speciality_Bucket_Endo/Onc/U
0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0
1	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0
2	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0
3	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0
4	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0

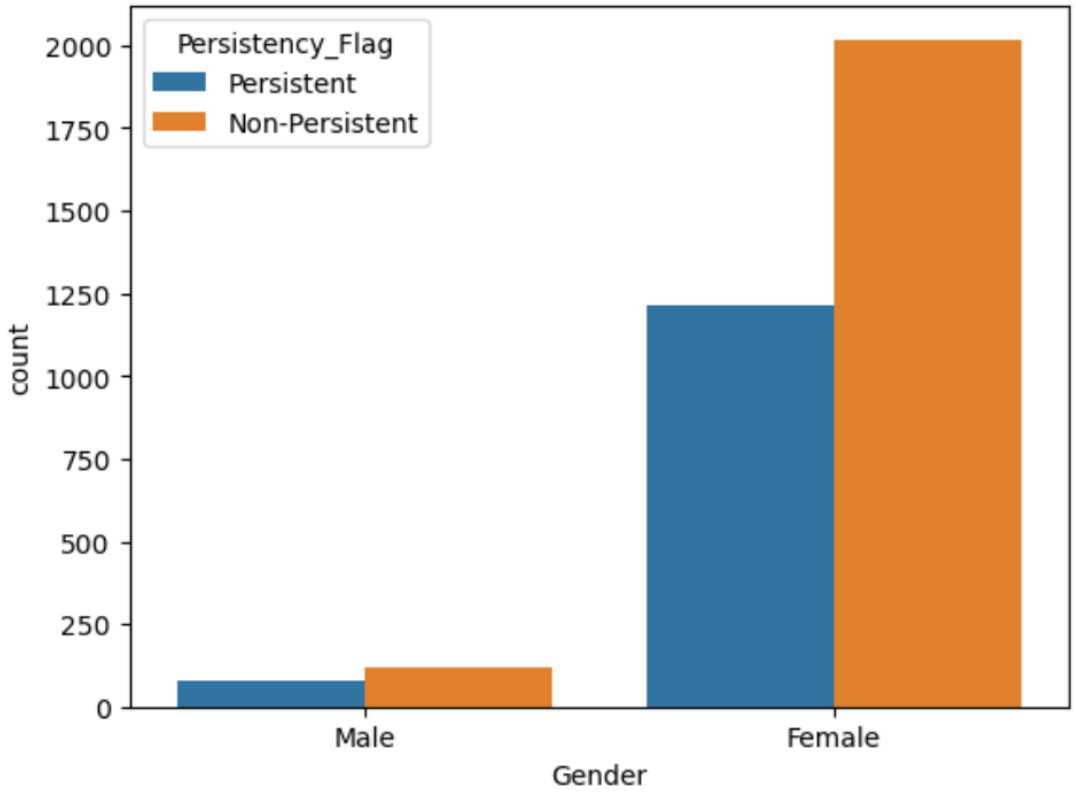
Code target variables

```python
# label encode other features
for col in bool_cols:
    label(df, col)
```

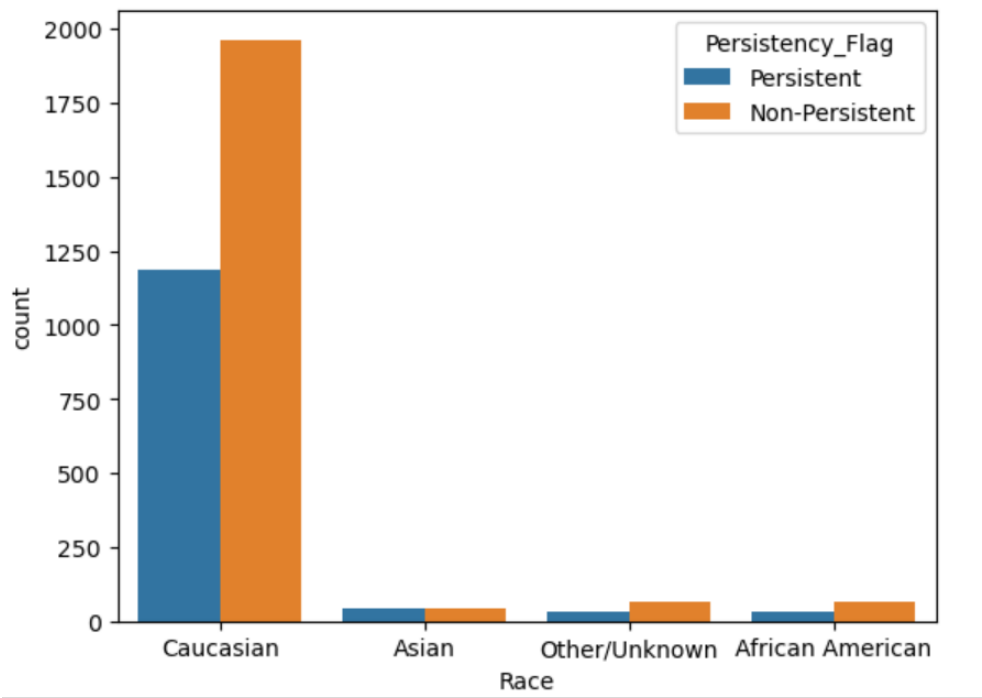
```python
# code target variables
df['target'] = np.where(df['Persistency_Flag'] == 'Persistent', 1, 0)
df['target']
```

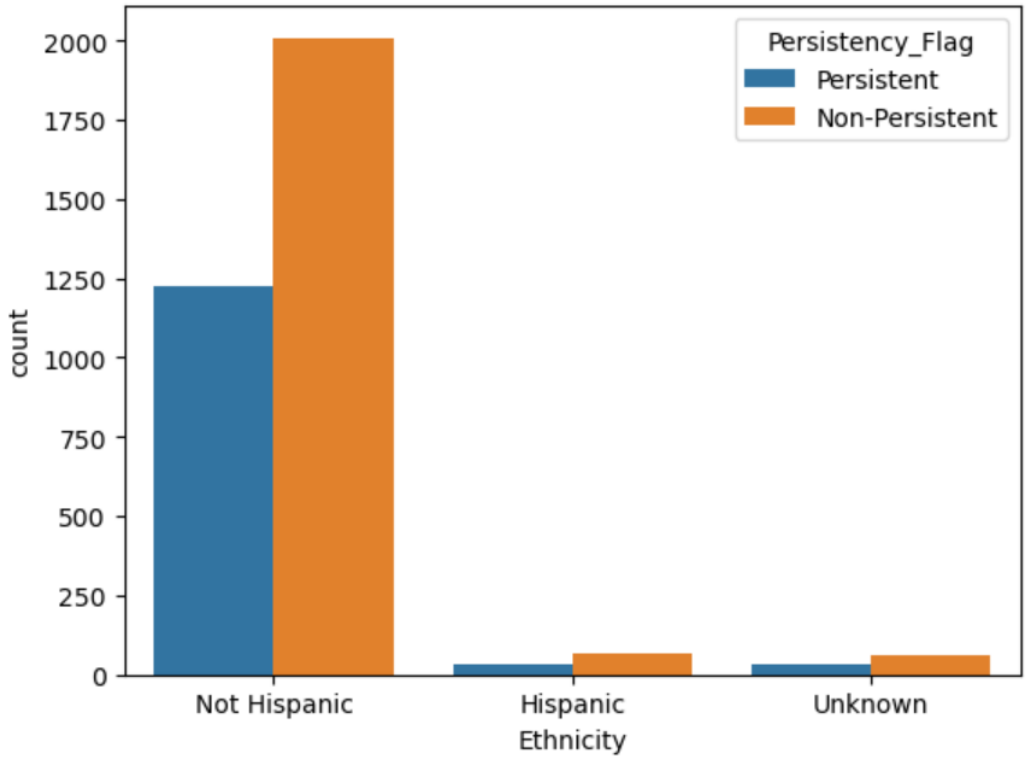
```
0       1
1       0
2       0
3       0
4       0
       ..
3419    1
3420    1
3421    1
3422    0
3423    0
Name: target, Length: 3424, dtype: int32
```

Exploratory Data Analysis and Recommendations



There is a huge distinction between the number of Male and Female patients, we will recommend not use this feature or weight it less.


Persistency_Flag
Persistent
Non-Persistent
2000
1750
1500
1250
1000
750
500
250
0
count
Caucasian
Asian
Other/Unknown
African American
Race



Again, Race and Ethnicity groups are not balanced in the dataset, one way is that we can group other groups together into a new bigger group.

All the variables above have very unbalanced data distribution, as a result we recommend to group other values together into a new group for later model feature selection.

The healthcare dataset has many Boolean columns that contains YES or NO values. Again many of the unbalanced features will not be recommended for later use.

Model Selection and Building

Through the model-building process, I evaluated two Linear Classification models (Logistic Regression, Support Vector Machine), one Ensemble model (Random Forest), and one Boosting model (Extreme Gradient Boosting).

First, the dataset was split into train and test sets for model evaluation. Then, we use the training set to train fit the model, then take the test set to predict target variables. Finally, use different evaluation metrics to evaluate performance of models.

Logistic Regression:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
clr = LogisticRegressionCV(cv=5, scoring='accuracy')
```

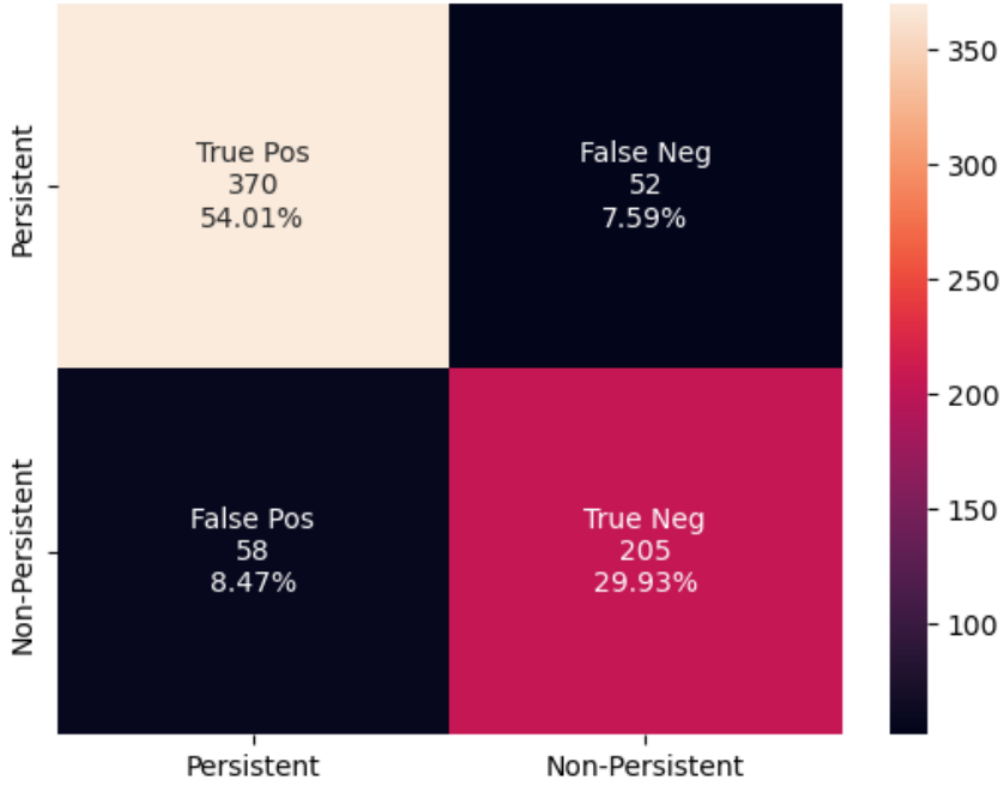
```
clr.fit(X_train, y_train)
```

```
LogisticRegressionCV(cv=5, scoring='accuracy')
```

```
pred = clr.predict(X_test)
```

```
acc_log = accuracy_score(y_test, pred)
print("Accuracy Score:", acc_log)
pre_log = precision_score(y_test, pred)
print('Precision Score:', pre_log)
recall_log = recall_score(y_test, pred)
print('Recall Score:', recall_log)
f1_log = f1_score(y_test, pred)
print('F1 Score:', f1_log)
roc_log = roc_auc_score(y_test, pred)
print('ROC Area:', roc_log)
```

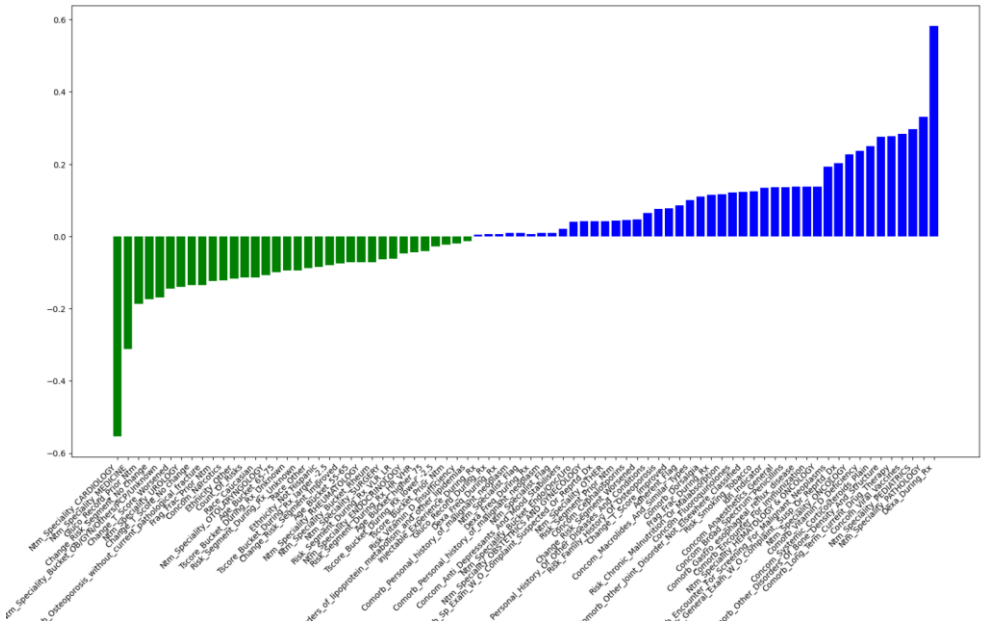
```
Accuracy Score: 0.8394160583941606
Precision Score: 0.8148148148148148
Recall Score: 0.752851711026616
F1 Score: 0.7826086956521738
ROC Area: 0.8231083199682843
```



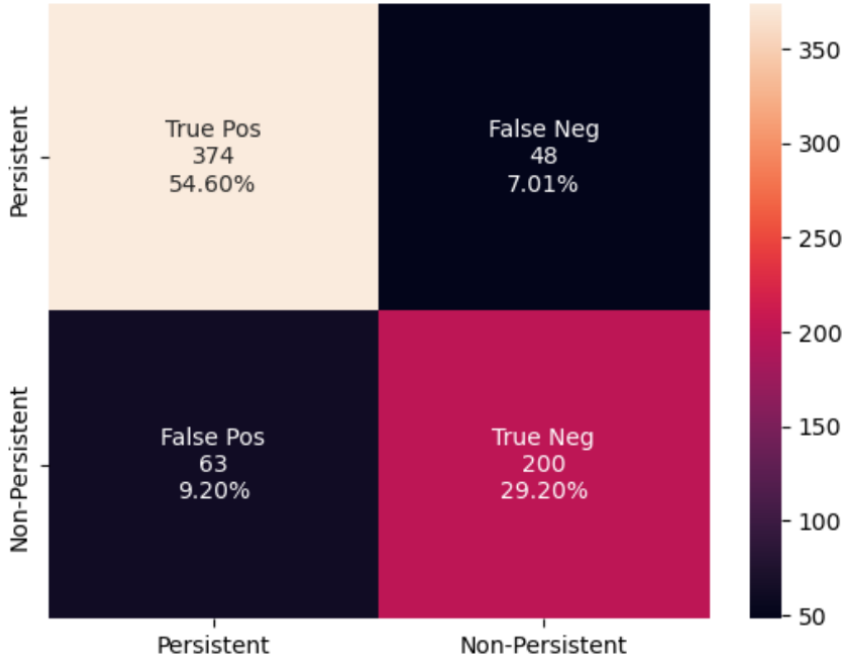
Linear SVC:

After fitting Linear SVC Model, I visualized all features by their importance level and remove the unimportant features:

```
# visualize feature importance
num_features = 39
coef = svc.coef_.ravel()
top_positive_coefficients = np.argsort(coef)[-num_features:]
top_negative_coefficients = np.argsort(coef)[:num_features]
top_coefficients = np.hstack([top_negative_coefficients, top_positive_coefficients])
plt.figure(figsize=(20,10))
colors = ['green' if c < 0 else 'blue' for c in coef[top_coefficients]]
plt.bar(np.arange(2 * num_features), coef[top_coefficients], color=colors)
feature_names = np.array(X.columns.values)
plt.xticks(np.arange( 2 * num_features), feature_names[top_coefficients], rotation=45, ha='right')
plt.show()
```

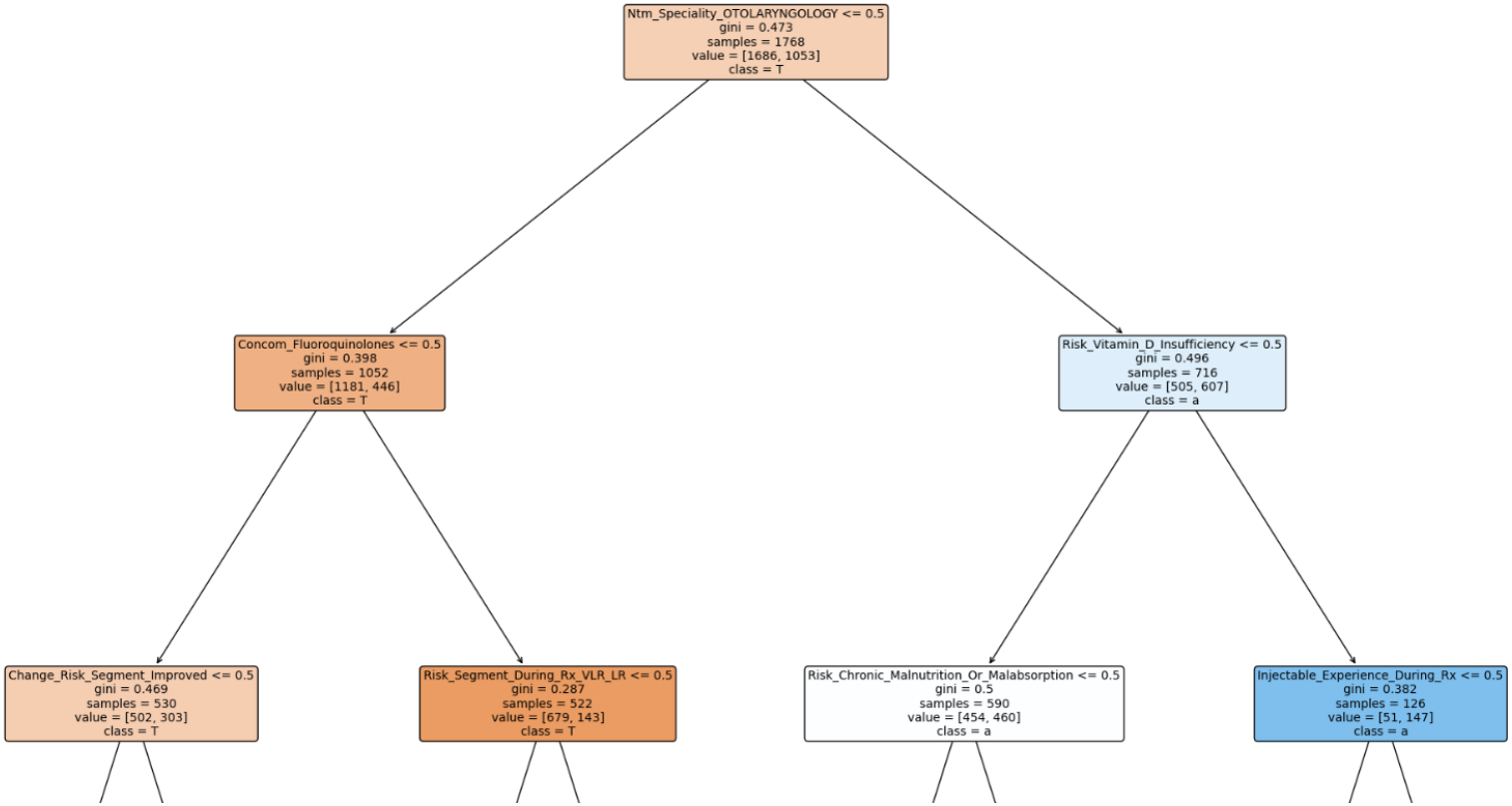


```
Accuracy Score: 0.8379562043795621
Precision Score: 0.8064516129032258
Recall Score: 0.7604562737642585
F1 Score: 0.7827788649706457
ROC Area: 0.8233560989674372
```

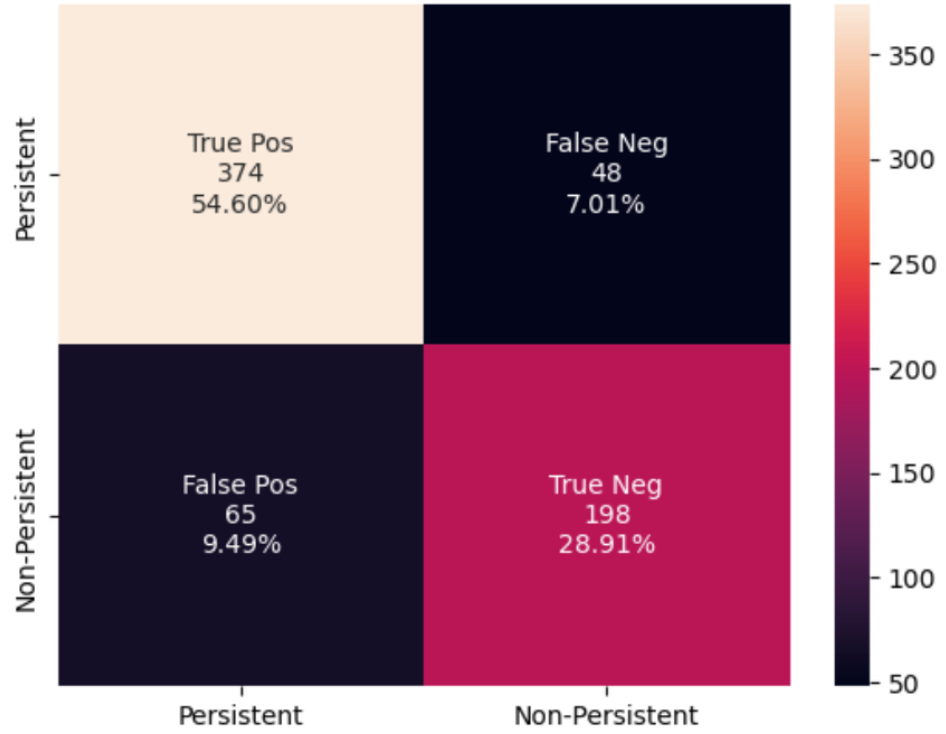


Random Forest:

Fit the data to Ensemble model Random Forest and visualized the top nodes of some of the decision trees:



```
Accuracy Score: 0.8350364963503649
Precision Score: 0.8048780487804879
Recall Score: 0.752851711026616
F1 Score: 0.7779960707269156
ROC Area: 0.8195538175986161
```



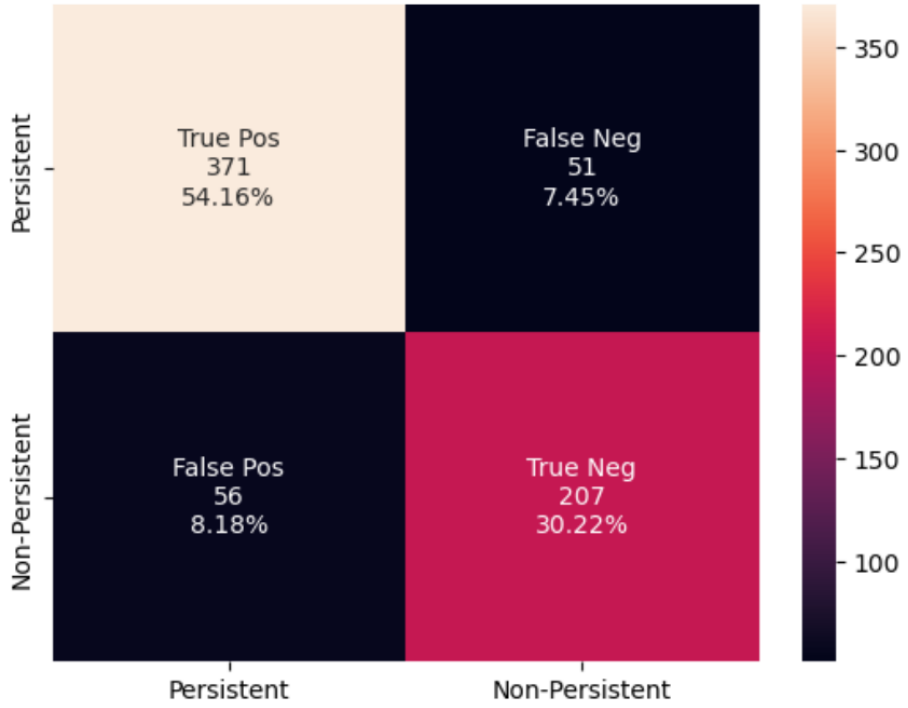
XGboost:

For Boosting model, I selected XGboost to fit the data. I used Bayesian optimization to tune the hyperparameters:

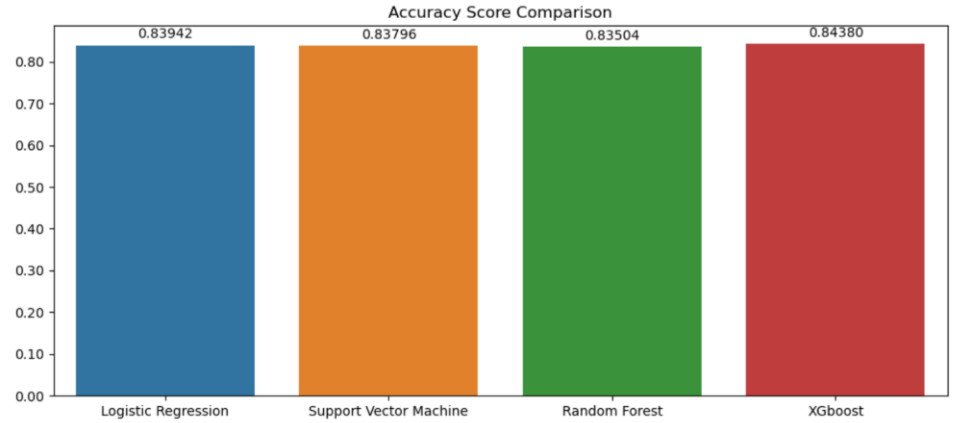
```
params_gbm ={
    'max_depth':(1, 10),
    'learning_rate':(0.01, 1),
    'n_estimators':(80, 500),
    'subsample': (0.8, 1),
}
gbm_bo = BayesianOptimization(gbm_cl_bo, params_gbm, random_state=1)
gbm_bo.maximize(init_points=20, n_iter=4)
```

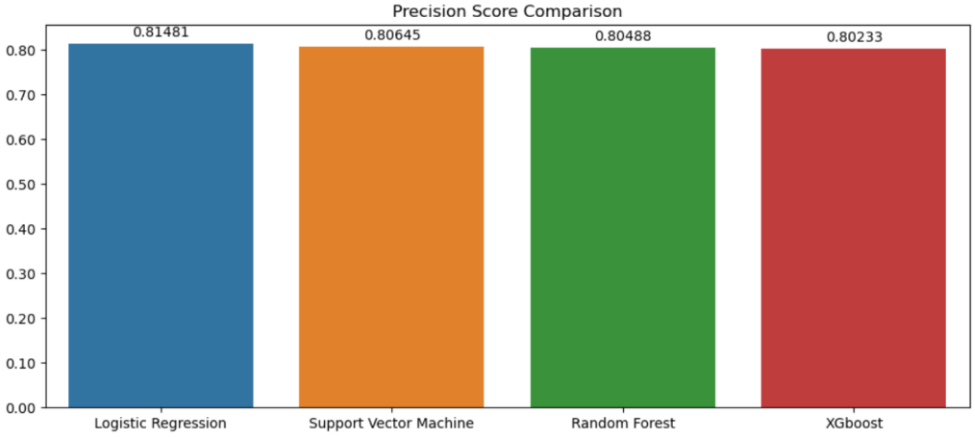
```
|   iter    |  target   | learni... | max_depth | n_esti... | subsample |
-------------------------------------------------------------------------
| 1         | 0.793     | 0.4229    | 7.483     | 80.05     | 0.8605    |
| 2         | 0.8109    | 0.1553    | 1.831     | 158.2     | 0.8691    |
| 3         | 0.7824    | 0.4028    | 5.849     | 256.1     | 0.937     |
| 4         | 0.7904    | 0.2124    | 8.903     | 91.5      | 0.9341    |
| 5         | 0.7857    | 0.4231    | 6.028     | 139.0     | 0.8396    |
| 6         | 0.789     | 0.8027    | 9.714     | 211.6     | 0.9385    |
| 7         | 0.782     | 0.8776    | 9.051     | 115.7     | 0.8078    |
| 8         | 0.797     | 0.1781    | 8.903     | 121.3     | 0.8842    |
| 9         | 0.7715    | 0.9583    | 5.798     | 370.6     | 0.8631    |
| 10        | 0.7839    | 0.6896    | 8.512     | 87.68     | 0.95      |
| 11        | 0.7802    | 0.989     | 7.733     | 197.8     | 0.9579    |
| 12        | 0.7915    | 0.1122    | 5.031     | 461.6     | 0.8587    |
| 13        | 0.8109    | 0.2949    | 2.17      | 88.13     | 0.9358    |
| 14        | 0.8003    | 0.2195    | 3.39      | 286.5     | 0.8107    |
| 15        | 0.7901    | 0.5784    | 2.321     | 327.5     | 0.94      |
| 16        | 0.7923    | 0.1113    | 4.727     | 371.6     | 0.8828    |
| 17        | 0.7967    | 0.05945   | 5.823     | 358.8     | 0.903     |
| 18        | 0.7882    | 0.9451    | 6.279     | 459.4     | 0.8275    |
| 19        | 0.7981    | 0.1479    | 8.267     | 247.0     | 0.8331    |
| 20        | 0.778     | 0.9282    | 4.13      | 395.3     | 0.9452    |
| 21        | 0.8076    | 0.01764   | 2.457     | 377.9     | 0.9529    |
| 22        | 0.7864    | 0.4481    | 4.721     | 310.1     | 0.9488    |
| 23        | 0.7956    | 0.3617    | 8.191     | 247.1     | 0.9915    |
| 24        | 0.8098    | 0.09787   | 1.855     | 159.1     | 0.8381    |
=========================================================================
```

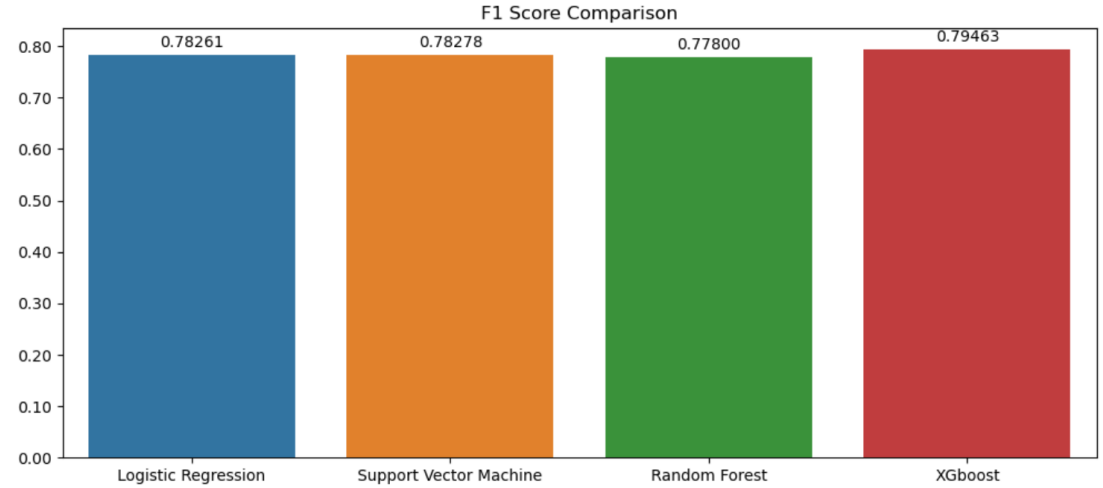
Accuracy Score: 0.8437956204379562
Precision Score: 0.8023255813953488
Recall Score: 0.7870722433460076
F1 Score: 0.7946257197696737
ROC Area: 0.8331095813886435

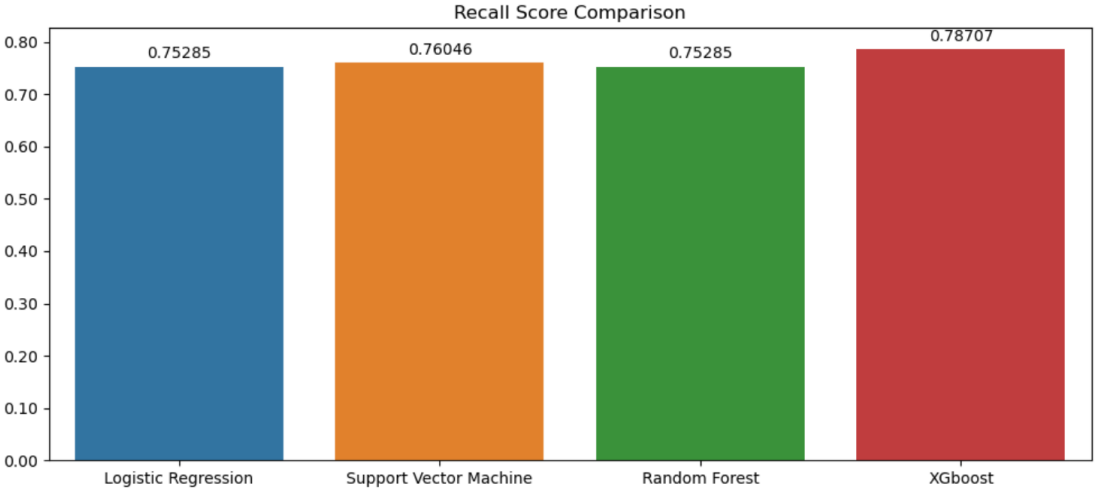


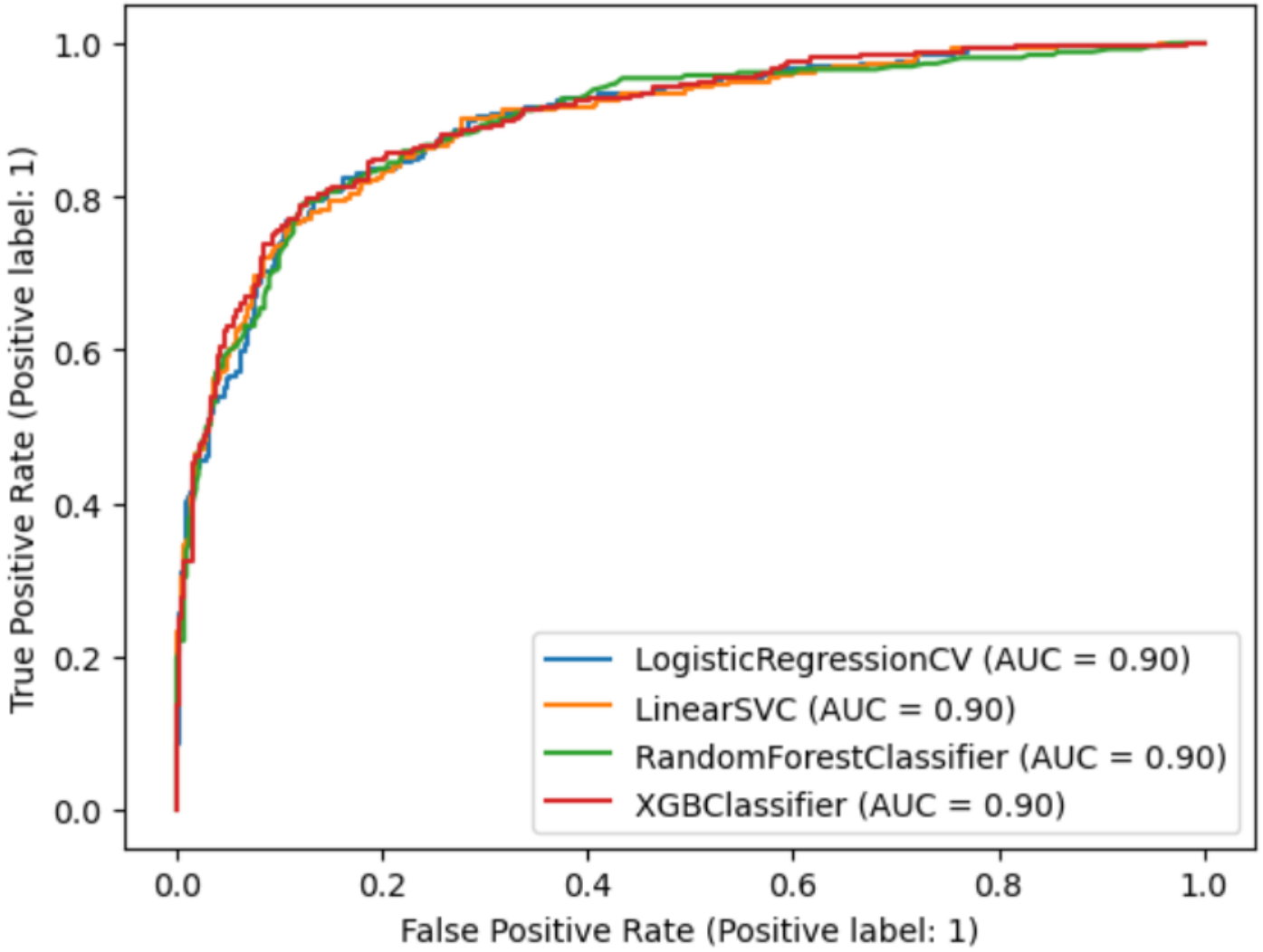
I then grouped all the performances together into barplots to compare the difference:






F1 Score Comparison
0.78261
0.78278
0.77800
0.79463
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
Logistic Regression
Support Vector Machine
Random Forest
XGboost


Recall Score Comparison
0.75285
0.76046
0.75285
0.78707
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
Logistic Regression
Support Vector Machine
Random Forest
XGboost


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)
LogisticRegressionCV (AUC = 0.90)
LinearSVC (AUC = 0.90)
RandomForestClassifier (AUC = 0.90)
XGBClassifier (AUC = 0.90)

Overall, every model has pretty similar performance regarding different evaluation metrics. Among them, XGboost has the best average performance. So it is recommended to use the XGboost model to make predictions. The model is save as the best_model file for later use.

```
import pickle
```

```
# save the best model
filename = 'best_model.sav'
pickle.dump(clf, open(filename, 'wb'))
```

```
loaded_model = pickle.load(open(filename, 'rb'))
loaded_model.score(X_test, y_test)
```

```
0.8437956204379562
```