PROBLEM 1: Patient "Target Drug" Eligibility Prediction

INTRODUCTION:

We're building predictive models to forecast if a patient can use 'Target Drug' 30 days in advance. This helps doctors choose the right treatment. We'll check how good our models are using the F1-Score, which is like a measure of how well they work.

OBJECTIVE:

- Develop a predictive model that can determine whether a patient will be eligible for "Target Drug" treatment in the next 30 days.
- The model should be based on patient data such as demographics, medical history, and current health status.
- The goal is to help physicians make informed decisions on which treatments to give to their patients.In this case, certain patients have been identified as having taken the target drug and are therefore classified as the positive class. Now, we need to predict whether a patient will be eligible for that drug or not, 30 days in advance.

DATASET DESCRIPTION:

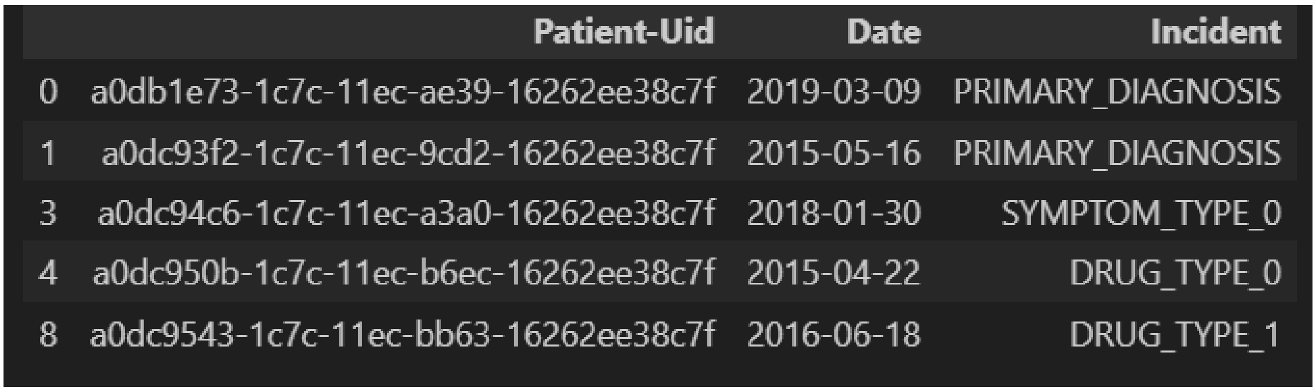
We have the data in the form of below 2 files

- Train.parquet - Dataset to be used for training
- Test.parquet - Dataset to be used for testing

The dataset in question contains a comprehensive collection of electronic health records belonging to patients who have been diagnosed with a specific disease. These health records comprise a detailed log of every aspect of the patients' medical history, including all diagnoses, symptoms, prescribed drug treatments, and medical tests that they have undergone. Each row represents a healthcare record/medical event for a patient and it includes a timestamp for each entry/event, thereby allowing for a chronological view of the patient's medical history.\

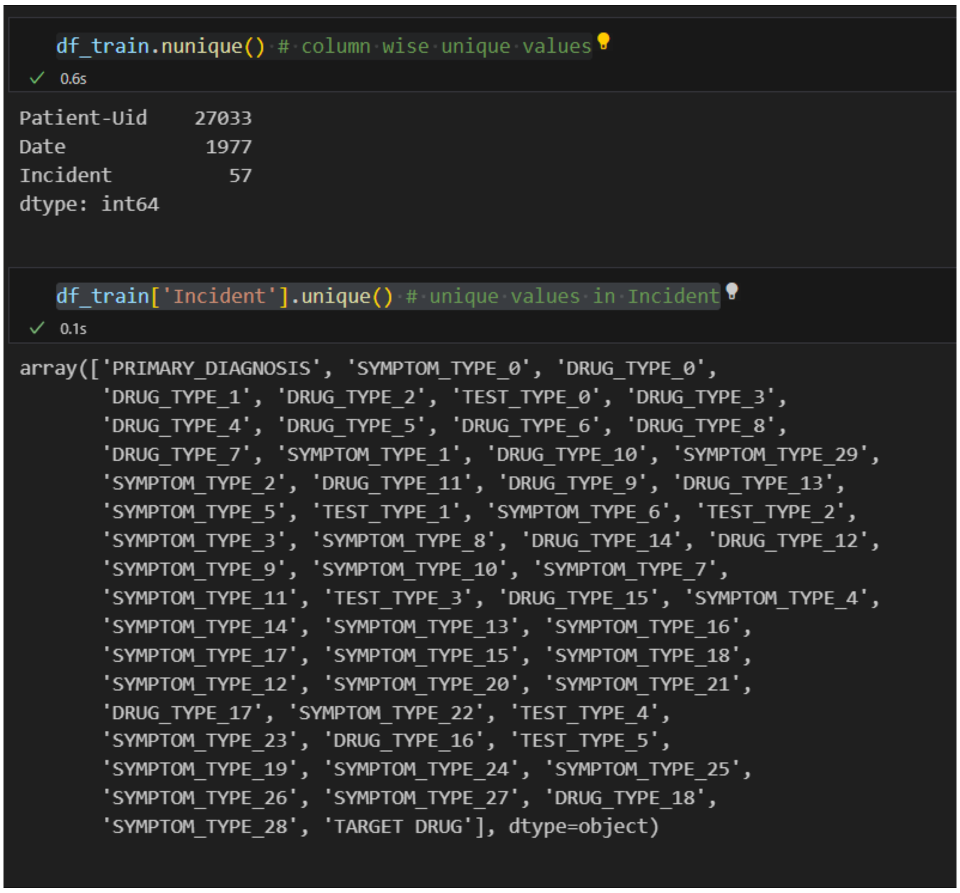
The Data has mainly three columns:

- **Patient-Uid** - Unique Alphanumeric Identifier for a patient
- **Date** - Date when patient encountered the event.
- **Incident** – This columns describes which event occurred on the day.



	Patient-Uid	Date	Incident
0	a0db1e73-1c7c-11ec-ae39-16262ee38c7f	2019-03-09	PRIMARY_DIAGNOSIS
1	a0dc93f2-1c7c-11ec-9cd2-16262ee38c7f	2015-05-16	PRIMARY_DIAGNOSIS
3	a0dc94c6-1c7c-11ec-a3a0-16262ee38c7f	2018-01-30	SYMPTOM_TYPE_0
4	a0dc950b-1c7c-11ec-b6ec-16262ee38c7f	2015-04-22	DRUG_TYPE_0
8	a0dc9543-1c7c-11ec-bb63-16262ee38c7f	2016-06-18	DRUG_TYPE_1

All the data is anonymized to protect the privacy of the patients and hence incidents are of type may be drugs, tests or any other activity are categorical anonymous features.



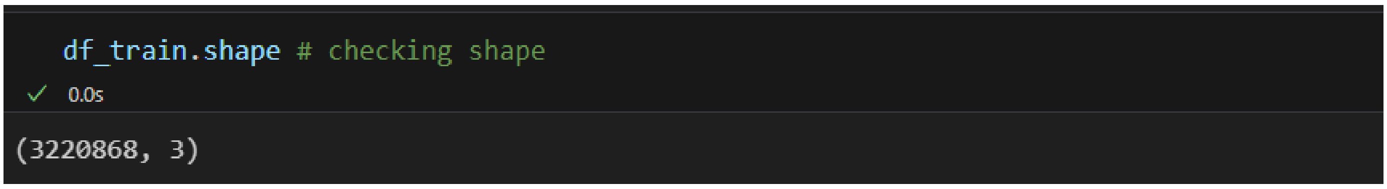
```
df_train.nunique() # column wise unique values
✓ 0.6s

Patient-Uid    27033
Date            1977
Incident          57
dtype: int64
```

```
df_train['Incident'].unique() # unique values in Incident
✓ 0.1s

array(['PRIMARY_DIAGNOSIS', 'SYMPTOM_TYPE_0', 'DRUG_TYPE_0',
       'DRUG_TYPE_1', 'DRUG_TYPE_2', 'TEST_TYPE_0', 'DRUG_TYPE_3',
       'DRUG_TYPE_4', 'DRUG_TYPE_5', 'DRUG_TYPE_6', 'DRUG_TYPE_8',
       'DRUG_TYPE_7', 'SYMPTOM_TYPE_1', 'DRUG_TYPE_10', 'SYMPTOM_TYPE_29',
       'SYMPTOM_TYPE_2', 'DRUG_TYPE_11', 'DRUG_TYPE_9', 'DRUG_TYPE_13',
       'SYMPTOM_TYPE_5', 'TEST_TYPE_1', 'SYMPTOM_TYPE_6', 'TEST_TYPE_2',
       'SYMPTOM_TYPE_3', 'SYMPTOM_TYPE_8', 'DRUG_TYPE_14', 'DRUG_TYPE_12',
       'SYMPTOM_TYPE_9', 'SYMPTOM_TYPE_10', 'SYMPTOM_TYPE_7',
       'SYMPTOM_TYPE_11', 'TEST_TYPE_3', 'DRUG_TYPE_15', 'SYMPTOM_TYPE_4',
       'SYMPTOM_TYPE_14', 'SYMPTOM_TYPE_13', 'SYMPTOM_TYPE_16',
       'SYMPTOM_TYPE_17', 'SYMPTOM_TYPE_15', 'SYMPTOM_TYPE_18',
       'SYMPTOM_TYPE_12', 'SYMPTOM_TYPE_20', 'SYMPTOM_TYPE_21',
       'DRUG_TYPE_17', 'SYMPTOM_TYPE_22', 'TEST_TYPE_4',
       'SYMPTOM_TYPE_23', 'DRUG_TYPE_16', 'TEST_TYPE_5',
       'SYMPTOM_TYPE_19', 'SYMPTOM_TYPE_24', 'SYMPTOM_TYPE_25',
       'SYMPTOM_TYPE_26', 'SYMPTOM_TYPE_27', 'DRUG_TYPE_18',
       'SYMPTOM_TYPE_28', 'TARGET DRUG'], dtype=object)
```

There are **27033 unique patients** and **57 different types of Incident** or events present in the training dataset



```
df_train.shape # checking shape
✓ 0.0s

(3220868, 3)
```

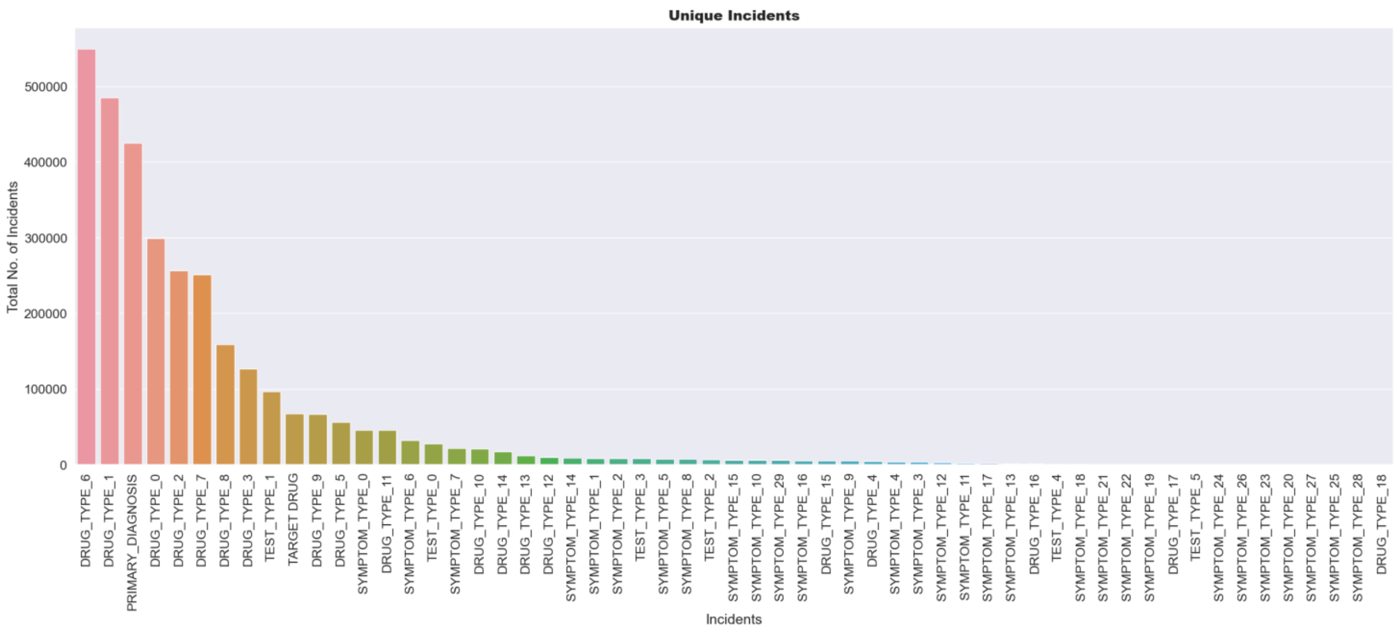
This shows that there are close to **3 million records** of the patients at our hand, and all the patients records are mixed so we need to separate them ourselves.

There is a drug called '**Target Drug'**, this drug is of interest, there are total **9K patients** in training data who have taken "Target Drug" at least once.

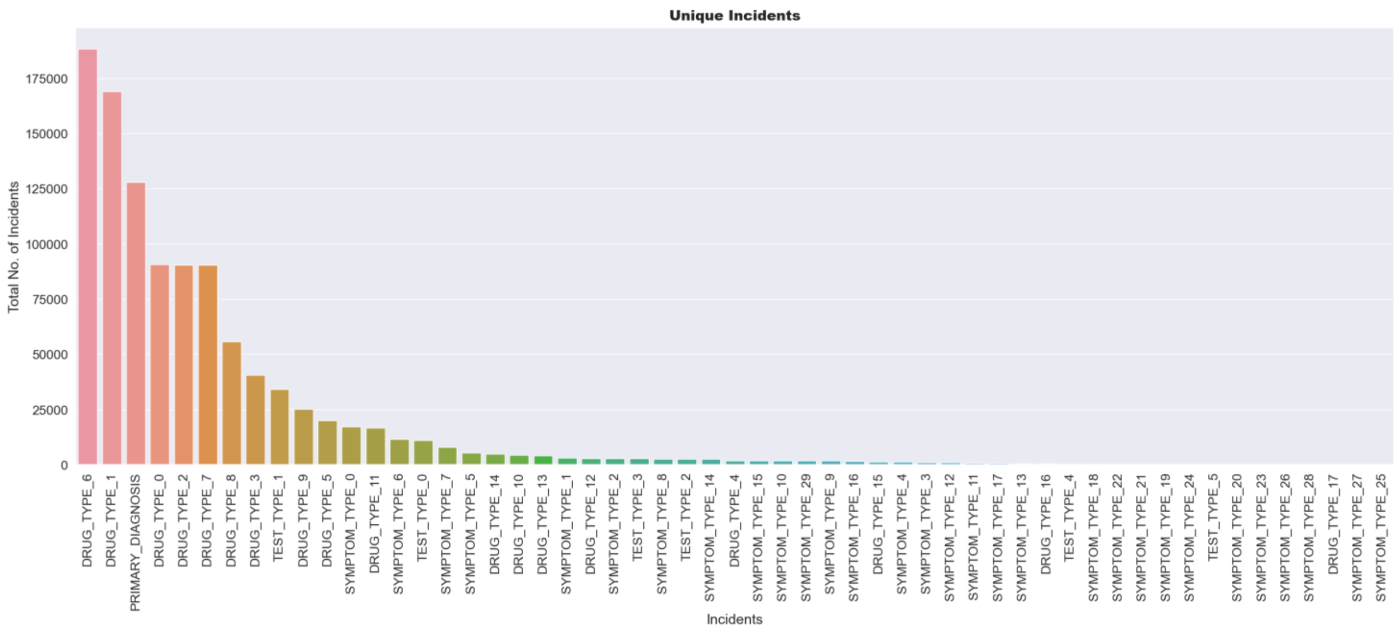
There is NO Duplicate values present in the train data.

There are **12,100** Duplicate values present in the test data.

Visualising Incident column in testing data



Visualising Incident column in testing data



SPLITTING DATASET:

Splitting dataset into two types. The following are the two types:

- Positive Data
- Negative Data

Positive Data:

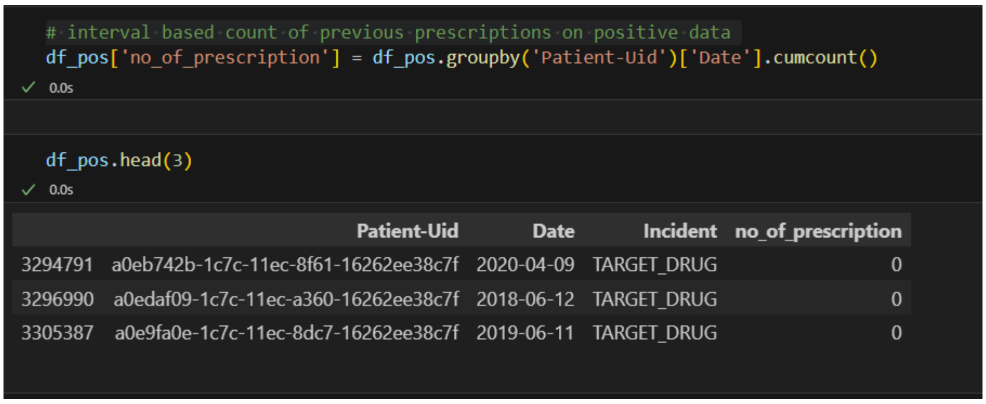
- Data Frame which includes target drug are positive which arerepresented as 1

Negative Data:

- Data Frame which excludes target drug are negative which are represented as 0

N0_OF_PRESCRIPTION COLUMN

- This code calculates the count of previous prescriptions within intervals for the 'df_neg' dataframe.



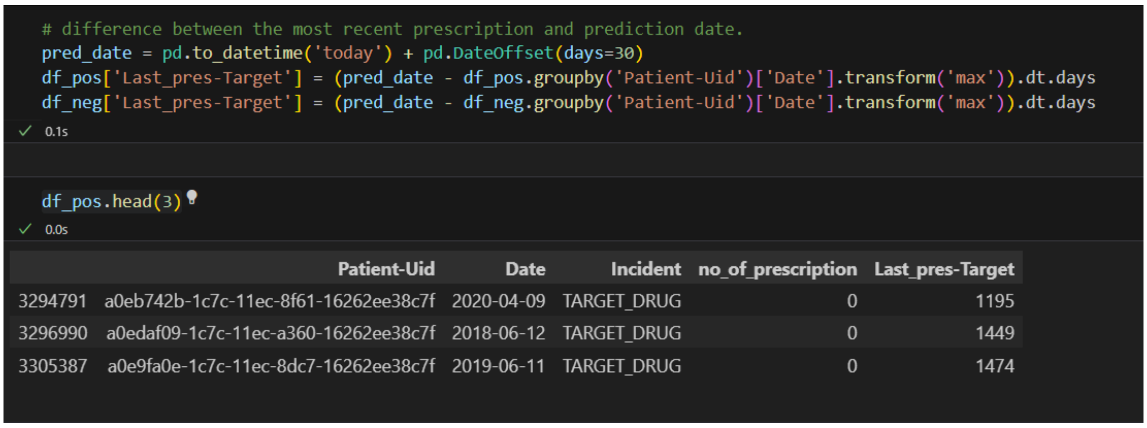
```
# interval based count of previous prescriptions on positive data
df_pos['no_of_prescription'] = df_pos.groupby('Patient-Uid')['Date'].cumcount()
```
✓ 0.0s

```
df_pos.head(3)
```
✓ 0.0s

	Patient-Uid	Date	Incident	no_of_prescription
3294791	a0eb742b-1c7c-11ec-8f61-16262ee38c7f	2020-04-09	TARGET_DRUG	0
3296990	a0edaf09-1c7c-11ec-a360-16262ee38c7f	2018-06-12	TARGET_DRUG	0
3305387	a0e9fa0e-1c7c-11ec-8dc7-16262ee38c7f	2019-06-11	TARGET_DRUG	0

Last_pres-Target

- Created a new column called 'Last_pres-Target'. Calculate the difference in days between the target prediction date and the most recent prescription date for each unique 'Patient-Uid' group in the 'Date' column.
- This code calculates the number of days between the most recent prescription date and a target prediction date for two dataframes: df_pos and df_neg.



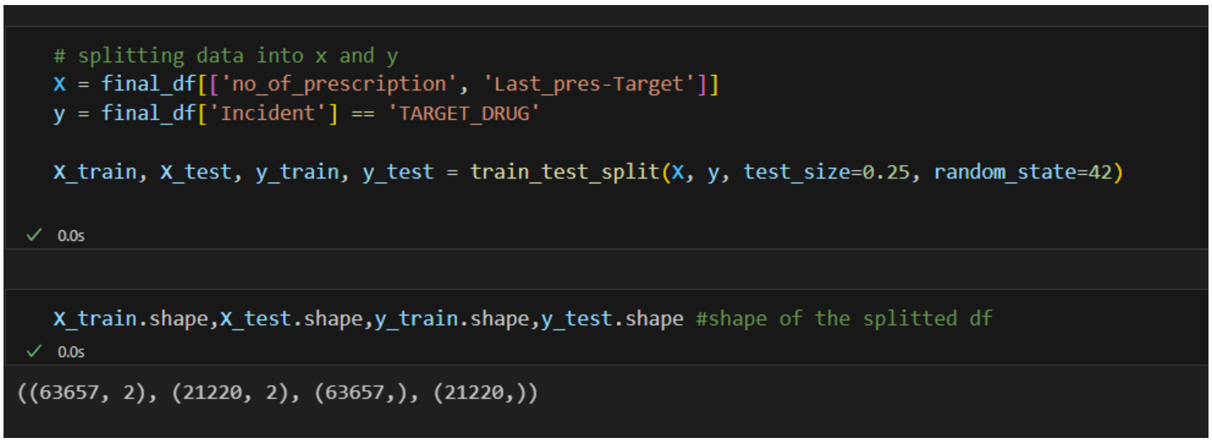
```
# difference between the most recent prescription and prediction date.
pred_date = pd.to_datetime('today') + pd.DateOffset(days=30)
df_pos['Last_pres-Target'] = (pred_date - df_pos.groupby('Patient-Uid')['Date'].transform('max')).dt.days
df_neg['Last_pres-Target'] = (pred_date - df_neg.groupby('Patient-Uid')['Date'].transform('max')).dt.days
```
✓ 0.1s

```
df_pos.head(3)
```
✓ 0.0s

	Patient-Uid	Date	Incident	no_of_prescription	Last_pres-Target
3294791	a0eb742b-1c7c-11ec-8f61-16262ee38c7f	2020-04-09	TARGET_DRUG	0	1195
3296990	a0edaf09-1c7c-11ec-a360-16262ee38c7f	2018-06-12	TARGET_DRUG	0	1449
3305387	a0e9fa0e-1c7c-11ec-8dc7-16262ee38c7f	2019-06-11	TARGET_DRUG	0	1474

Splitting Data into Predictors (X) and Target (Y)

This code is preparing the data for machine learning by splitting it into predictor variables (X) and the target variable (Y). It then further divides the data into training and testing sets.



```
# splitting data into x and y
X = final_df[['no_of_prescription', 'Last_pres-Target']]
y = final_df['Incident'] == 'TARGET_DRUG'

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```
✓ 0.0s

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape #shape of the splitted df
```
✓ 0.0s

```
((63657, 2), (21220, 2), (63657,), (21220,))
```

Building Models:

Model Selection:

Three distinct machine learning models were selected for this project to solve a classification problem.

- Logistic Regression
- Random Forest Classifier
- XGBoost Classifier.

The primary evaluation metric employed for this model is the F1 score.

Model Descriptions:

Logistic Regression:

- Logistic Regression is a simple yet effective algorithm for binary classification.

- It's suitable for cases where you need to predict if an item belongs to one category or another.

Random Forest Classifier:

- Random Forest Classifier is an ensemble learning technique based on decision trees.

- It can handle various types of classification problems and can be quite robust.

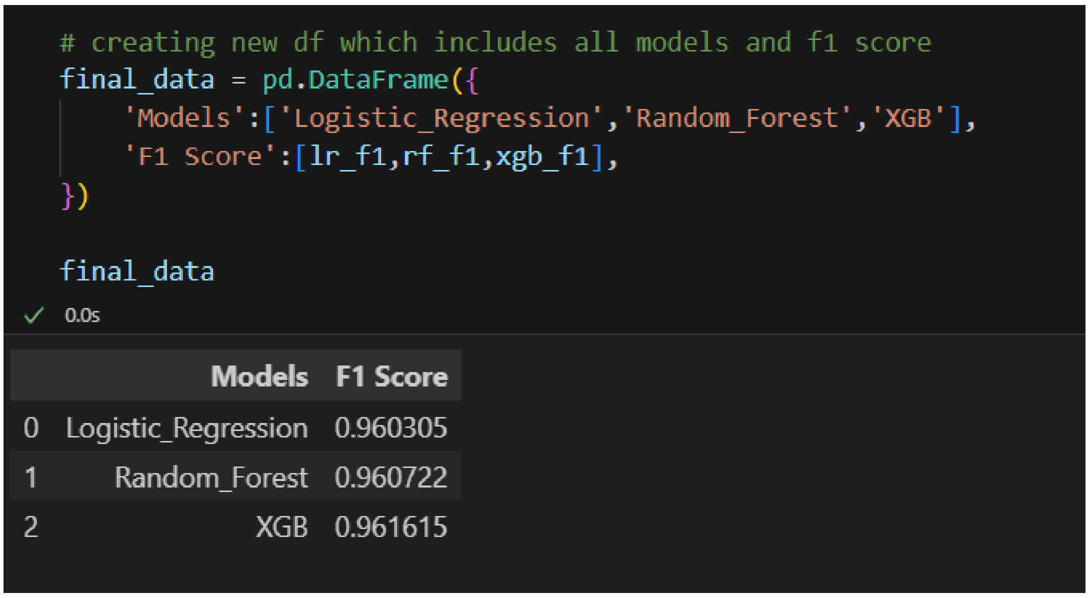
XGBoost Classifier:

- XGBoost Classifier is a powerful gradient boosting algorithm.

- It's known for high predictive accuracy and can be used for complex classification tasks.

Model Evaluation - F1 Score:

- This code creates a new Data Frame named final_data to compare the performance of different machine learning models based on their F1 scores.



```
# creating new df which includes all models and f1 score
final_data = pd.DataFrame({
    'Models':['Logistic_Regression','Random_Forest','XGB'],
    'F1 Score':[lr_f1,rf_f1,xgb_f1],
})

final_data
```

```
✓ 0.0s

                Models  F1 Score
0  Logistic_Regression  0.960305
1        Random_Forest  0.960722
2                  XGB  0.961615
```

Logistic Regression: Achieved an F1 score of approximately **0.960**, indicating a strong balance between precision and recall.

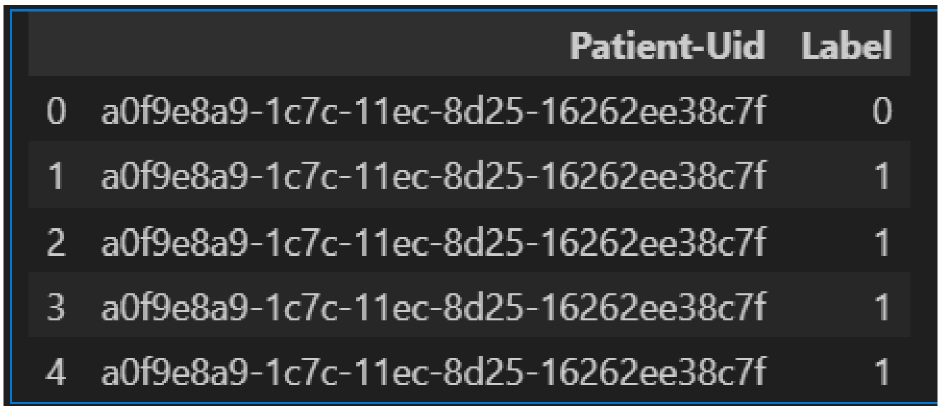
Random Forest Classifier: Delivered an F1 score of around **0.961**, showing excellent performance in classification tasks.

XGBoost Classifier: Outperformed the other models with an F1 score of about **0.962**, indicating high accuracy and reliability in predictions.

The XGBoost Classifier emerged as the top-performing model in this evaluation, but all three models demonstrated impressive results.

Final Submission File (Test Data):

- This section outlines the process of generating the final submission file, which includes predicted labels for test data. The example provided uses a DataFrame named `Test_output_df`.
- The `Test_output_df` DataFrame serves as the foundation for the final submission file, which includes 'Patient-Uid' and 'Label' columns. The 'Label' column contains the predicted values, and this DataFrame can be saved as a CSV file for submission, completing the model's predictions for the test dataset.



	Patient-Uid	Label
0	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	0
1	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	1
2	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	1
3	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	1
4	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	1

FUTURE ENHANCEMENT (If I have more time):

1**. Feature Engineering**: I would have to Implement feature engineering techniques, including time-based and frequency-based features, to enhance the model's F1 score. Time-based features can capture temporal patterns, while frequency-based features can distinguish between patients.

2. **Deep Learning Techniques**: I would have considered using deep learning approaches, specifically Long Short-Term Memory (LSTM) networks, given the time-based nature of the data. LSTM models are well-suited for capturing temporal dependencies.

3**. Feature Tokenization**: Incorporate feature tokenization as part of the encoding process, which can improve the model's ability to understand and utilize the data effectively.

4. **Addressing Data Imbalance**: I would have explored oversampling techniques to address imbalanced data, ensuring a more balanced and representative dataset for model training and evaluation.