Structured Data Assignment – Akaike

By Kavibarathi K

Problem statement:

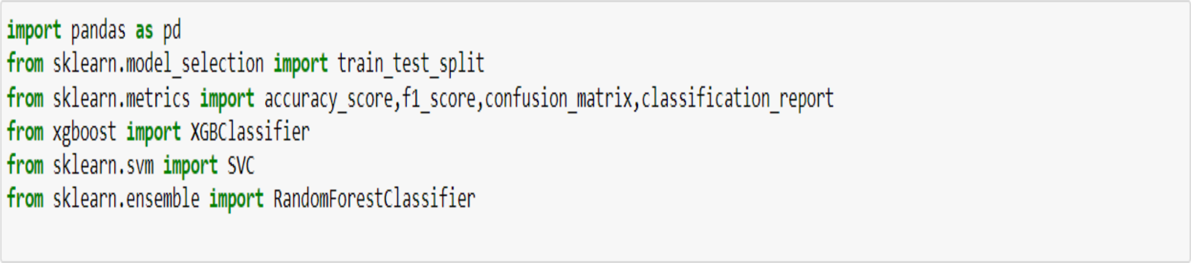
The development of drugs is critical in providing therapeutic options for patients suffering from chronic and terminal illnesses. “Target Drug”, in particular, is designed to enhance the patient's health and well-being without causing dependence on other medications that could potentially lead to severe and life-threatening side effects. These drugs are specifically tailored to treat a particular disease or condition, offering a more focused and effective approach to treatment, while minimising the risk of harmful reactions.

Objective:

The objective in this assignment is to develop a predictive model which will predict whether a patient will be eligible*** for “Target Drug” or not in next 30 days. Knowing if the patient is eligible or not will help physician treating the patient make informed decision on the which treatments to give.

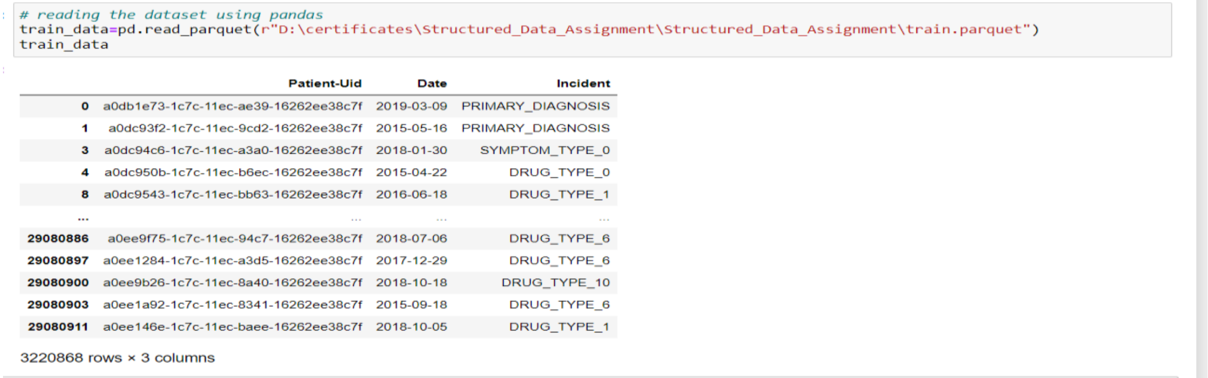
Necessary Libraries:

These are python libraries that are necessary to perform the functions for the model predictions.



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,f1_score,confusion_matrix,classification_report
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

Reading the Training Dataset:



```
# reading the dataset using pandas
train_data=pd.read_parquet(r"D:\certificates\Structured_Data_Assignment\Structured_Data_Assignment\train.parquet")
train_data
```

	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
...	...	...	...
29080886	a0ee9f75-1c7c-11ec-94c7-16262ee38c7f	2018-07-06	DRUG_TYPE_6
29080897	a0ee1284-1c7c-11ec-a3d5-16262ee38c7f	2017-12-29	DRUG_TYPE_6
29080900	a0ee9b26-1c7c-11ec-8a40-16262ee38c7f	2018-10-18	DRUG_TYPE_10
29080903	a0ee1a92-1c7c-11ec-8341-16262ee38c7f	2015-09-18	DRUG_TYPE_6
29080911	a0ee146e-1c7c-11ec-baee-16262ee38c7f	2018-10-05	DRUG_TYPE_1

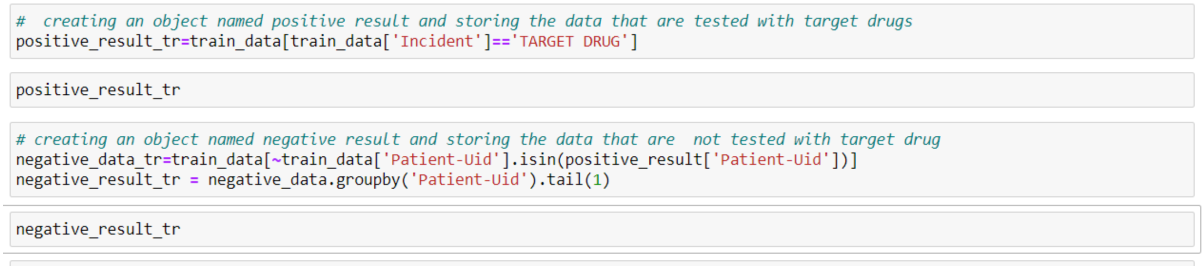
3220868 rows × 3 columns

Data Preprocessing:

This step includes detailed analysis of dataset and perform necessary process like checking for missing values , data type conversions , removing duplicate data’s in the dataset. This is the most important process prior to model building in the predictions.

Creating a Positive and Negative data frames:

In this the positive data contains the set of data that are being test using the target drug and in the negative data incidents that are other than the “target drug” are taken into account .



```
#  creating an object named positive result and storing the data that are tested with target drugs
positive_result_tr=train_data[train_data['Incident']=='TARGET DRUG']
```

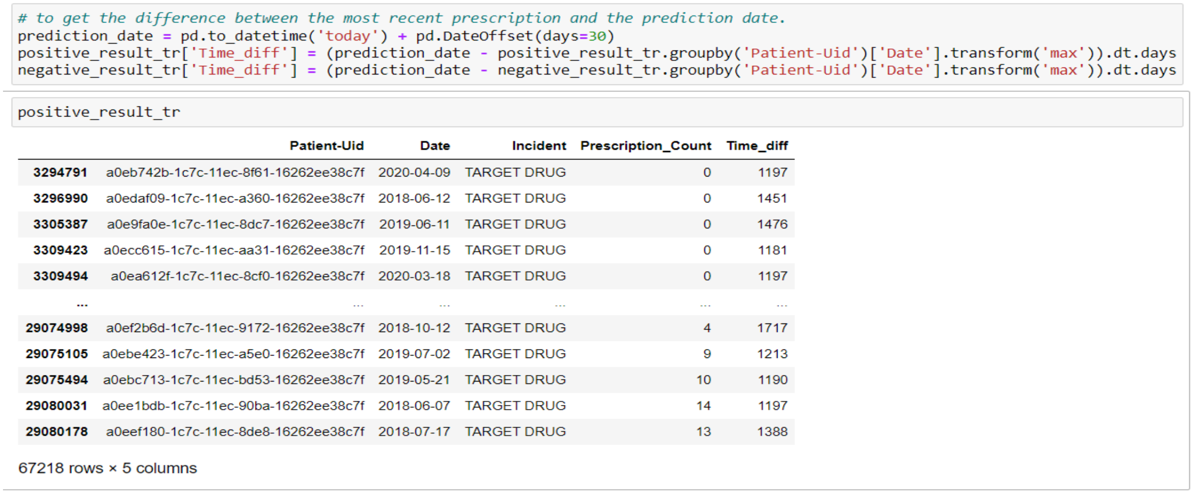
```
positive_result_tr
```

```
# creating an object named negative result and storing the data that are  not tested with target drug
negative_data_tr=train_data[~train_data['Patient-Uid'].isin(positive_result['Patient-Uid'])]
negative_result_tr = negative_data.groupby('Patient-Uid').tail(1)
```

```
negative_result_tr
```

Time difference column:

In this column, find out the time taken for the prescriptions of drug to find out the symptoms.



```
# to get the difference between the most recent prescription and the prediction date.
prediction_date = pd.to_datetime('today') + pd.DateOffset(days=30)
positive_result_tr['Time_diff'] = (prediction_date - positive_result_tr.groupby('Patient-Uid')['Date'].transform('max')).dt.days
negative_result_tr['Time_diff'] = (prediction_date - negative_result_tr.groupby('Patient-Uid')['Date'].transform('max')).dt.days
```

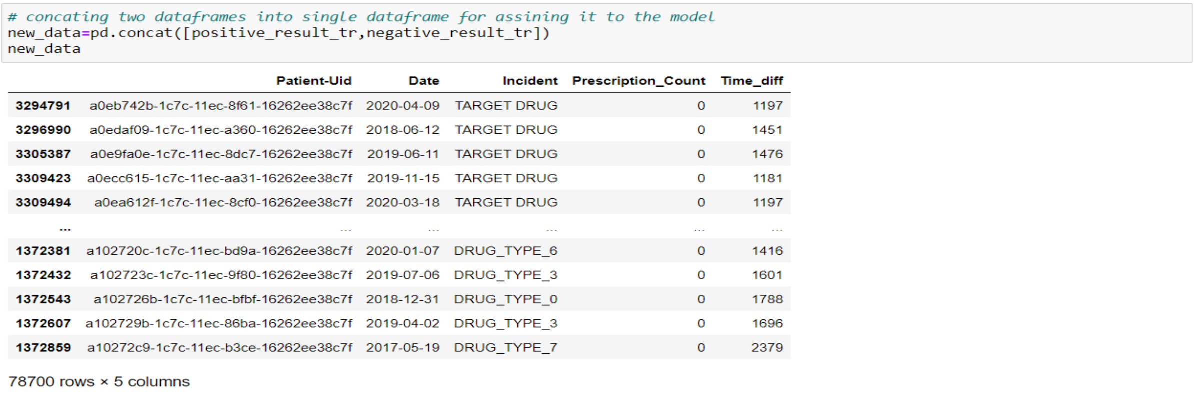
```
positive_result_tr
```

	Patient-Uid	Date	Incident	Prescription_Count	Time_diff
3294791	a0eb742b-1c7c-11ec-8f61-16262ee38c7f	2020-04-09	TARGET DRUG	0	1197
3296990	a0edaf09-1c7c-11ec-a360-16262ee38c7f	2018-06-12	TARGET DRUG	0	1451
3305387	a0e9fa0e-1c7c-11ec-8dc7-16262ee38c7f	2019-06-11	TARGET DRUG	0	1476
3309423	a0ecc615-1c7c-11ec-aa31-16262ee38c7f	2019-11-15	TARGET DRUG	0	1181
3309494	a0ea612f-1c7c-11ec-8cf0-16262ee38c7f	2020-03-18	TARGET DRUG	0	1197
...	...	...	...	...	...
29074998	a0ef2b6d-1c7c-11ec-9172-16262ee38c7f	2018-10-12	TARGET DRUG	4	1717
29075105	a0ebe423-1c7c-11ec-a5e0-16262ee38c7f	2019-07-02	TARGET DRUG	9	1213
29075494	a0ebc713-1c7c-11ec-bd53-16262ee38c7f	2019-05-21	TARGET DRUG	10	1190
29080031	a0ee1bdb-1c7c-11ec-90ba-16262ee38c7f	2018-06-07	TARGET DRUG	14	1197
29080178	a0eef180-1c7c-11ec-8de8-16262ee38c7f	2018-07-17	TARGET DRUG	13	1388

67218 rows × 5 columns

Concatenation the dataset:

The positive and negative dataset’s are joined together using the pandas concat function to perform the model predictions.



```
# concating two dataframes into single dataframe for assining it to the model
new_data=pd.concat([positive_result_tr,negative_result_tr])
new_data
```

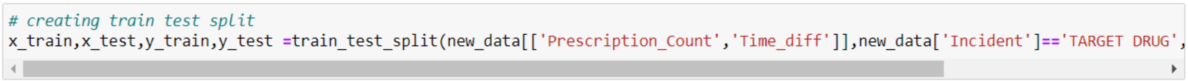
	Patient-Uid	Date	Incident	Prescription_Count	Time_diff
3294791	a0eb742b-1c7c-11ec-8f61-16262ee38c7f	2020-04-09	TARGET DRUG	0	1197
3296990	a0edaf09-1c7c-11ec-a360-16262ee38c7f	2018-06-12	TARGET DRUG	0	1451
3305387	a0e9fa0e-1c7c-11ec-8dc7-16262ee38c7f	2019-06-11	TARGET DRUG	0	1476
3309423	a0ecc615-1c7c-11ec-aa31-16262ee38c7f	2019-11-15	TARGET DRUG	0	1181
3309494	a0ea612f-1c7c-11ec-8cf0-16262ee38c7f	2020-03-18	TARGET DRUG	0	1197
...	...	...	...	...	...
1372381	a102720c-1c7c-11ec-bd9a-16262ee38c7f	2020-01-07	DRUG_TYPE_6	0	1416
1372432	a102723c-1c7c-11ec-9f80-16262ee38c7f	2019-07-06	DRUG_TYPE_3	0	1601
1372543	a102726b-1c7c-11ec-bfbf-16262ee38c7f	2018-12-31	DRUG_TYPE_0	0	1788
1372607	a102729b-1c7c-11ec-86ba-16262ee38c7f	2019-04-02	DRUG_TYPE_3	0	1696
1372859	a10272c9-1c7c-11ec-b3ce-16262ee38c7f	2017-05-19	DRUG_TYPE_7	0	2379

78700 rows × 5 columns

Model building:

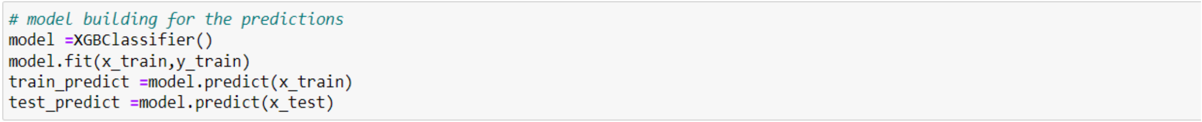
- The model building is the process in which we can use several classification algorithms to get the max accuracy from the dataset we have developed.

- Train-Test-Split:



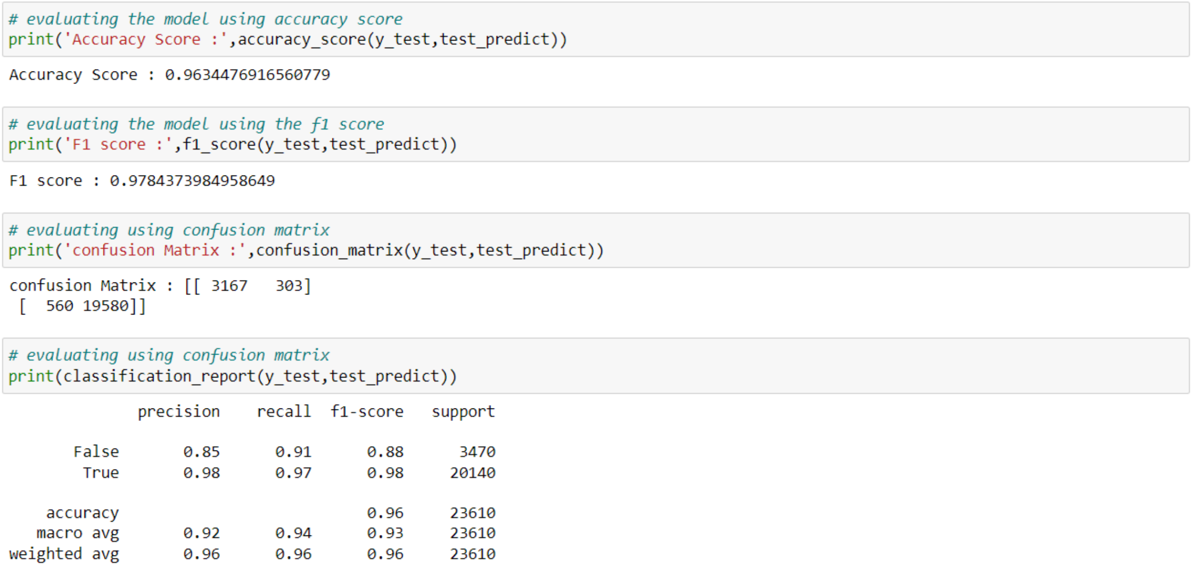
```
# creating train test split
x_train,x_test,y_train,y_test =train_test_split(new_data[['Prescription_Count','Time_diff']],new_data['Incident']=='TARGET DRUG',
```

- Model creation:



```
# model building for the predictions
model =XGBClassifier()
model.fit(x_train,y_train)
train_predict =model.predict(x_train)
test_predict =model.predict(x_test)
```

- Out of several model's that are tested for the best predictions XGBoost Classifier gave the highest F1-score.
- Model performance evaluated using several metrics.



```
# evaluating the model using accuracy score
print('Accuracy Score :',accuracy_score(y_test,test_predict))
```

```
Accuracy Score : 0.9634476916560779
```

```
# evaluating the model using the f1 score
print('F1 score :',f1_score(y_test,test_predict))
```

```
F1 score : 0.9784373984958649
```

```
# evaluating using confusion matrix
print('confusion Matrix :',confusion_matrix(y_test,test_predict))
```

```
confusion Matrix : [[ 3167   303]
 [  560 19580]]
```

```
# evaluating using confusion matrix
print(classification_report(y_test,test_predict))
```

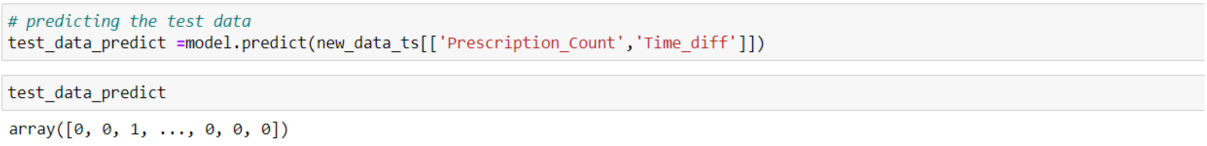
```
              precision    recall  f1-score   support

       False       0.85      0.91      0.88      3470
        True       0.98      0.97      0.98     20140

    accuracy                           0.96     23610
   macro avg       0.92      0.94      0.93     23610
weighted avg       0.96      0.96      0.96     23610
```

Tested model using test dataset:

In this data set all kind of necessary process as did in the training dataset.

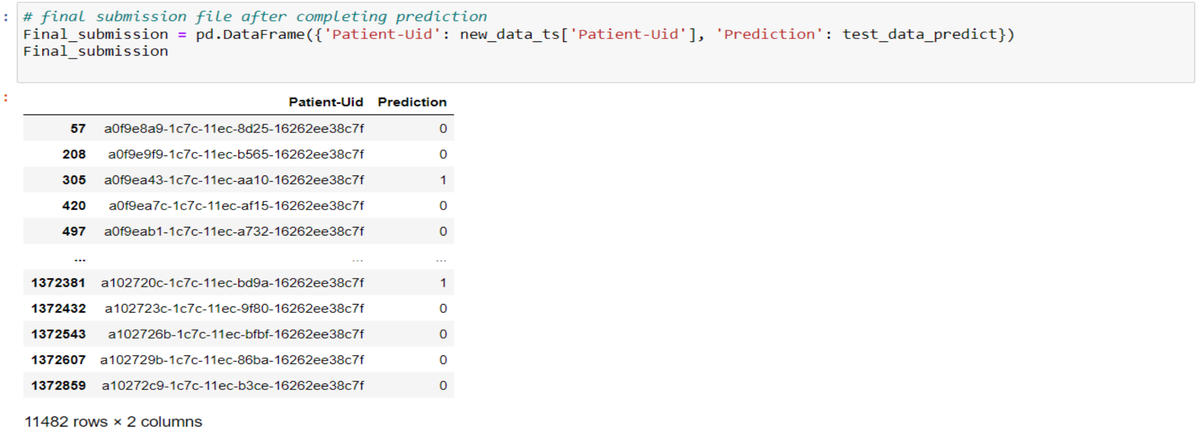


```
# predicting the test data
test_data_predict =model.predict(new_data_ts[['Prescription_Count','Time_diff']])
```

```
test_data_predict
```

```
array([0, 0, 1, ..., 0, 0, 0])
```

Final submission:



```
# final submission file after completing prediction
Final_submission = pd.DataFrame({'Patient-Uid': new_data_ts['Patient-Uid'], 'Prediction': test_data_predict})
Final_submission
```

	Patient-Uid	Prediction
57	a0f9e8a9-1c7c-11ec-8d25-16262ee38c7f	0
208	a0f9e9f9-1c7c-11ec-b565-16262ee38c7f	0
305	a0f9ea43-1c7c-11ec-aa10-16262ee38c7f	1
420	a0f9ea7c-1c7c-11ec-af15-16262ee38c7f	0
497	a0f9eab1-1c7c-11ec-a732-16262ee38c7f	0
...	...	...
1372381	a102720c-1c7c-11ec-bd9a-16262ee38c7f	1
1372432	a102723c-1c7c-11ec-9f80-16262ee38c7f	0
1372543	a102726b-1c7c-11ec-bfbf-16262ee38c7f	0
1372607	a102729b-1c7c-11ec-86ba-16262ee38c7f	0
1372859	a10272c9-1c7c-11ec-b3ce-16262ee38c7f	0

11482 rows × 2 columns