Healthcare Provider Fraud Detection

Group Members

Sahil Mallick

Srijani Adhikary

Mentor : Shubhadeep Das

[Senior Analyst at HSBC]

INTRODUCTION

Healthcare Provider Fraud is one of the biggest problems in today's world. According to the government, the total Medicare spending increased exponentially due to frauds in Medicare claims. Healthcare fraud is an organized crime which involves peers of providers, physicians, beneficiaries acting together to make fraud claims.

Healthcare fraud and abuse take many forms. Some of the most common types of frauds by providers are:

a) Billing for services that were not provided.

b) Duplicate submission of a claim for the same service.

c) Misrepresenting the service provided.

d) Charging for a more complex or expensive service than was actually provided.

e) Billing for a covered service when the service actually provided was not covered.

The goal of this project is to " predict the potentially fraudulent providers " based on the claims filed by them.

DATA DESCRIPTION

Here we are provided with 4 separate datasets:

A) <u>Inpatient Data</u>

This data provides insights about the claims filed for those patients who are admitted in the hospitals. It also provides additional details like their admission and discharge dates and admit diagnosis code.

- This dataset contains 40474 rows and 30 columns.

B) <u>Outpatient Data</u>

This data provides details about the claims filed for those patients who visit hospitals and not admitted in it.

- This dataset contains 517737 rows and 27 columns.

C) <u>Beneficiary Details Data</u>

This data contains beneficiary KYC details like health conditions, region they belong to etc.

- This dataset contains 138556 rows and 25 columns.

D) <u>Provider Data</u>

This data gives the value 1 or 0 against each provider depending on whether the Provider is fraud or not.

- This dataset contains 5410 rows and 2 columns.

The biggest challenge of this project is to prepare the data to make it suitable to train the models. First 3 datasets are not in given in Provider level. Beneficiary Data is at beneficiary level and IN & OUT patient data are at claim id level. We have to combine all the datasets in such a way that all the information will be presented against the providers.
In the data of Inpatients and Outpatients there are many providers. Under each provider there is at least one beneficiary. Each beneficiary has at least one claim id. In the data claim ids are unique.

DATA PREPARATION

- We grouped the Inpatient Data by provider and aggregated the claim diagnosis codes by count and unique count, physicians by unique count and Insurance Claim Amount Reimbursed by sum, mean, minimum, maximum and standard deviation.
- Then we concat these new feature columns at Provider level and obtained a new dataframe 'df_in'.
- We repeated the same steps for Outpatient data and obtained the dataframe 'df_out'.
- In Beneficiary Data we created the variable 'age' using date of birth and date of death and created dummy variables for gender and race.
- We merged Inpatient and outpatient data on common columns by outer join and then merged it with beneficiary data on 'beneID' by inner join.

- Now in this merged data we created a new variable 'Flag_suspicious' which takes the value 1 if number of unique providers for a particular beneficiary is greater than 1, and 0 otherwise.
- Then we grouped the merged data with respect to provider and aggregated the variables by count, unique count, sum, mean etc. as applicable and in this way found a range of new features. For example, we got average insurance claim amount reimbursed for a provider id, total instances of a particular chronic condition under a provider id etc. All the details of this feature creation procedure can be found in the notebook.
- By concating all these new features we got a new dataframe ‘df_merged’.
- We merged the datasets ‘df_in’, ‘df_out’ and ‘df_merged’ on provider level by left join and imputed the missing values with 0. We dropped the features for which no description is available to avoid any kind of misinterpretation. Thus we obtained the dataset named ‘health’. Now onwards we will use this data as our main data.

Development And Validation Sample

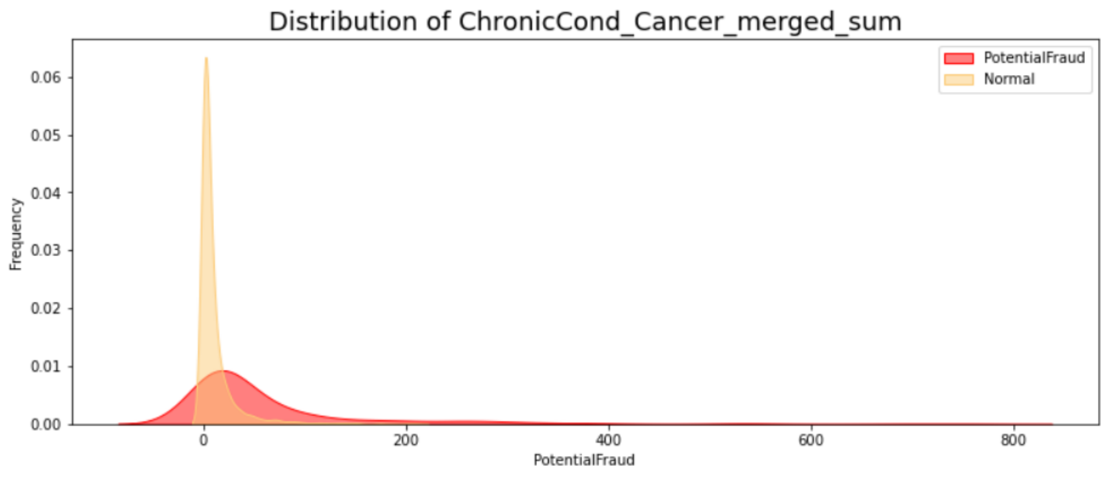
We split the provider data into 2 parts: development part (data1) on which we will build the model and validation part (data2) on which the model will be scored. Data1 and data2 contain 5004 rows and 406 rows respectively. Merge each of data1 and data 2 with ‘health’ on provider to make the development sample and validation sample respectively.

Deriving Features On The Development Sample

- We applied **Weight of Evidence Method** to bring all the features in same scale and further used those as our features instead of the original ones.

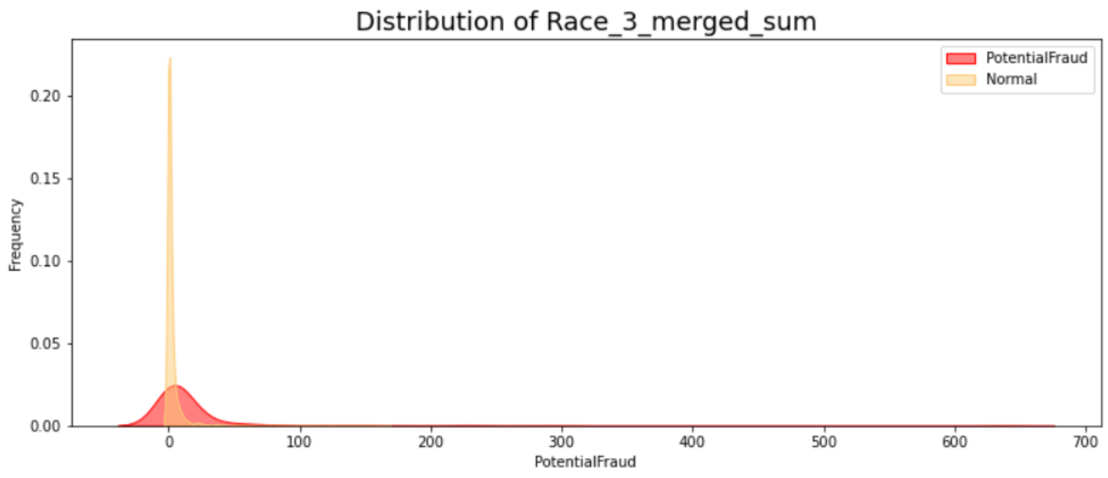
- We derived various binary flag variables from the existing variables by observing their ranges using kernel density plot. For example :

  - The feature 'derived_ChronicCond_Cancer_flag' takes the value 1 if total instances of Cancer disease under that Provider is >= 70



```
Value Counts of 1 and 0 :
0    116
1    111
Name: PotentialFraud, dtype: int64
Event Rate:48.90%
```

  - The feature 'derived_Race_3_flag' takes the value 1 if total no. of beneficiaries belonging to race 3 under a provider >= 30



```
Value Counts of 1 and 0 :
1    49
0    45
Name: PotentialFraud, dtype: int64
Event Rate:52.13%
```

- The feature ‘derived_InscClaimAmtReimbursed_merged_flag’ takes the value 1 if the maximum insurance claim amount of the merged data (‘df_merged) for that Provider is greater than the quantity mean+3*sigma of that variable.

  Other features created similarly can be found the notebook. We merged these flag variables with the features created by woe method to get the final list of features.

➢ Then we checked for correlation between the counts and unique counts of features created by WOE Method and found **high correlation (>0.9)** between them. So we removed one between count and unique count from each of them keeping that variable which **has more correlation with PotentialFraud.**

FEATURE SELECTION

We fitted **Decision Tree**, **Random Forest** and **CatBoost Classifier** to the development sample after the completion of feature creation part and obtained the quantitative values of importance of the features. We sorted them in decreasing order, observed the cumulative importance and considered the features upto that point where 80 percent of the total importance was crossed.

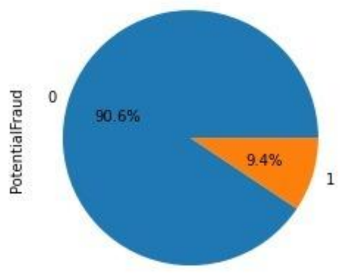
We also used the **weight of evidence** method to visualise the predictive power of the features by observing their **information value** and considered those with **IV** greater than 0.2.

Then we applied the **Recursive Feature Elimination** technique where we were able to rank all the features by setting the no. of features to be selected to 1. We used Random Forest classifier as the estimator here.

Then taking the ranks as pivot we selected those features which were considered to be important by at least 3 out of the remaining 4 methods. We ended up with 35 features out of which 2 are derived flag variables and rest are the features created by woe method.

MODEL TRAINING AND HYPERPARAMETER TUNING

We observed that the development sample is imbalanced with respect to potential fraud as shown in the figure:

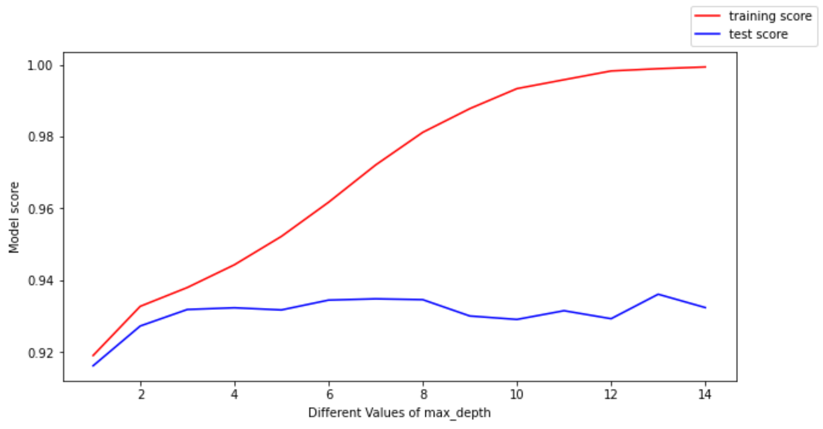


So we split the development sample with the selected features into train and test parts in 80:20 ratio by stratification and proceed for model fitting. We applied logistic regression, decision tree classifier, random forest classifier, lightgbm classifier and xgboost classifier. We observed that all the models except logistic regression were overfitting the data. So we performed hyperparameter tuning of those models with the help of bias-variance trade off.

➢ Bias-Variance Trade Off

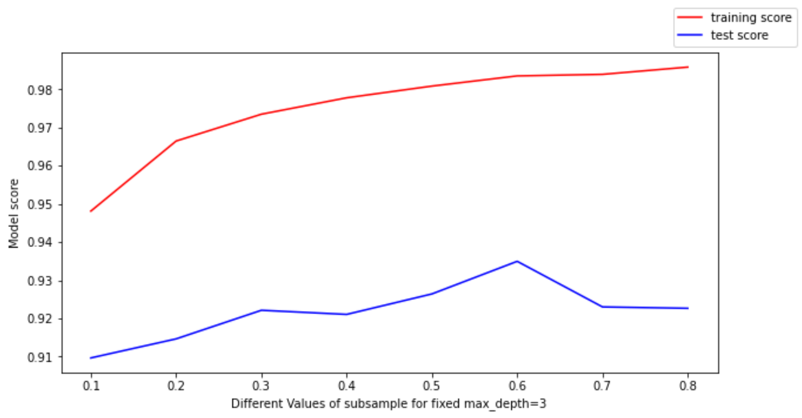
For building a good model we want both bias and variance to be low which is not possible to attain. So we need to find a proper balance between these two. Here we take a range of values of a particular parameter for a model and get an insight about the range where the bias and variance of the model are optimized.

For example, in case of Random Forest model the following graph shows model score against the parameter max_depth varying from 1 to 15.



This graph shows after value 3 the train AUC increases but test AUC decreases. So during hyperparameter tuning we varied max_depth from 2 to 5.

In another example , for the parameter subsample of xgboost model the following graph is obtained.



Observing the graph we decided to vary the parameter value from 0.2 to 0.6 during hyperparameter tuning.

Similarly we performed this bias variance trade off for some other parameters also in order to reduce overfitting and repeated this process for each of the models. Then we opted for hyperparameter tuning based on these results.

➢ Hyperparameter Tuning

Studying bias-variance trade off we performed a grid search method to get the best combination of parameters for all the models except logistic regression.

For example, for xgboost model we set the parameters as follows :

```
params_xgb = {'max_depth': [1,3,4,5,],
               'learning_rate':[0.025,0.03,0.05,0.055],
               'min_child_weight':[2,3,4],
               'n_estimators':[10,20,50,100],
               'subsample':[0.1,0.2,0.4,0.6]
              }
```

We got the best parameters :

```
'learning_rate': 0.03, 'max_depth': 3, 'min_child_weight': 3, 'n_estima
tors': 100, 'subsample': 0.4
```

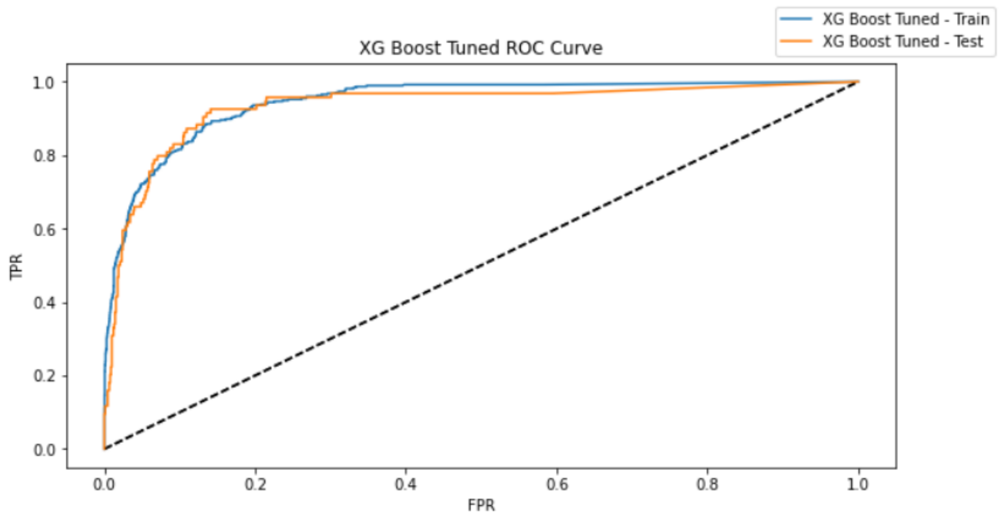
Selecting The Best Model

After getting the best combination of hyperparameters for each model we again fitted the models with those hyperparameter values on train and test parts. We checked the model performances on the basis of accuracy, precision, recall and AUC value. In case of our project identifying fraudulent providers is more important than identifying non-fraudulent cases. So we decided to choose the model with highest precision whose other performance measures are also high. Here we chose xgboost as our best model whose performance measures are given below:

```
========================== XG Boost Tuned - Model Report ==========================
===================== Model Accuracy ==============================
Train Accuracy: 0.939
Test Accuracy: 0.934
=====================================================
===================== Model Performance Metrices ===========================
Test Confusion Matrix:
[[886  21]
 [ 45  49]]
============================
Train Confusion Matrix:
[[3558   71]
 [ 172  202]]
=====================================================
Test Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       907
           1       0.70      0.52      0.60        94

    accuracy                           0.93      1001
   macro avg       0.83      0.75      0.78      1001
weighted avg       0.93      0.93      0.93      1001

=====================================================
Train Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.97      3629
           1       0.74      0.54      0.62       374

    accuracy                           0.94      4003
   macro avg       0.85      0.76      0.80      4003
weighted avg       0.93      0.94      0.93      4003

===================== Model AUC Scores ===========================
Train AUC Score: 0.945
Test AUC Score: 0.933
```

The ROC curve for this model is as follows :

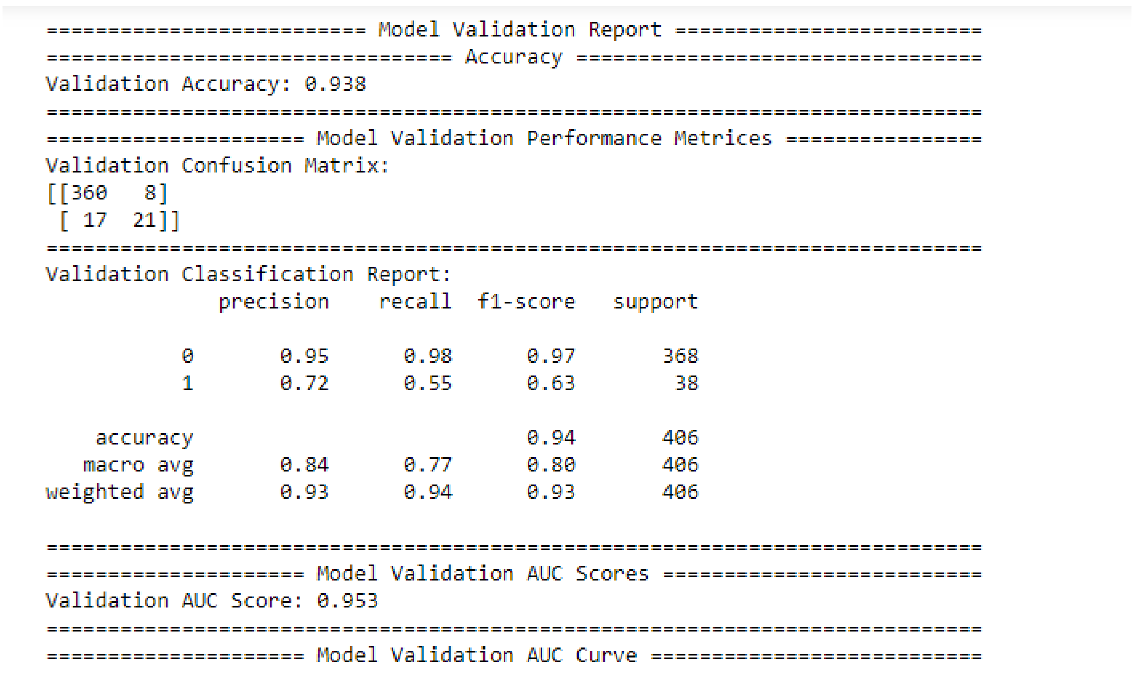
```
===================== Model AUC Scores ===========================
Train AUC Score: 0.945
Test AUC Score: 0.933
=====================================================
===================== Model AUC Curve ===========================
```



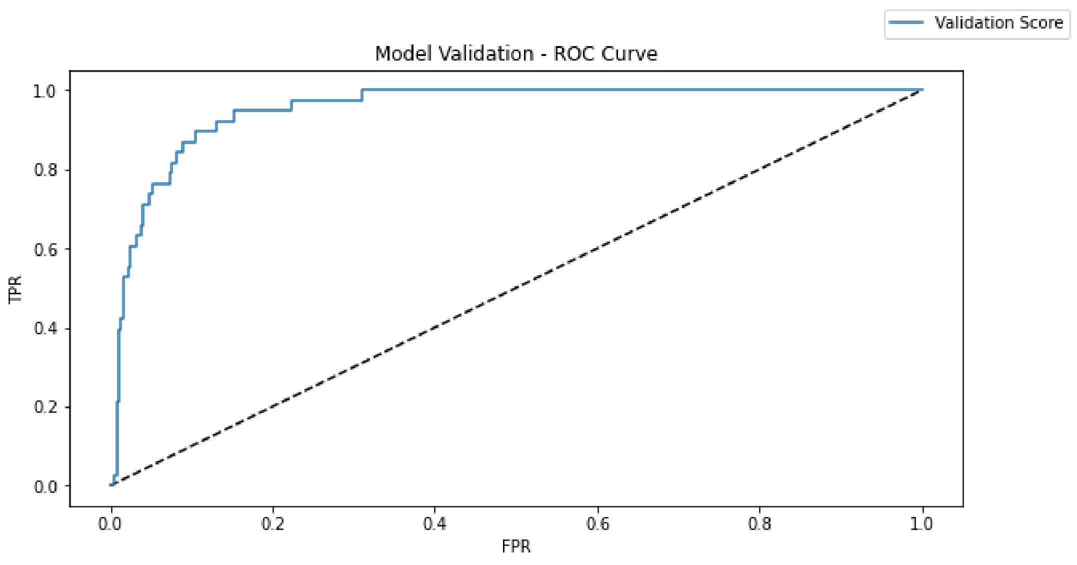
We saved this model in a pickle file for scoring purpose.

Scoring The Model on Validation Sample

Now on validation sample we created the features that were finally selected for model development in similar way. Then we scored the saved xgboost model on this data and got the following result :



```
========================= Model Validation Report ========================
================================ Accuracy ================================
Validation Accuracy: 0.938
==========================================================================
==================== Model Validation Performance Metrices ===============
Validation Confusion Matrix:
[[360   8]
 [ 17  21]]
==========================================================================
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.97       368
           1       0.72      0.55      0.63        38

    accuracy                           0.94       406
   macro avg       0.84      0.77      0.80       406
weighted avg       0.93      0.94      0.93       406

==========================================================================
==================== Model Validation AUC Scores =========================
Validation AUC Score: 0.953
==========================================================================
==================== Model Validation AUC Curve ==========================
```



The precision of this model is 0.72 which means the probability of identifying fraudulent providers correctly is 0.72 which is satisfactory for our purpose. The model accuracy and AUC are 0.938 and 0.953 which are quite high. So we can conclude that our model performs well on the validation sample and we can use this model for further fraud detection in thus field.

Interpretation of Important Features

We can conclude based on our observation that, while detecting fraudulent providers in the field of healthcare one should always keep in mind some key factors:

- **insurance claim amount reimbursed**: Our common sense tells that if for some provider the insurance claim amount reimbursed is very high, it may be suspicious.
- **age of the beneficiaries:** If ages of the beneficiaries under a provider are very high or very low it may be suspicious.
- **instances of various diseases:** If a provider files claims for a particular disease frequently it may be a fraudulent activity.
- **physician in charge:** If a physician's name is more frequent in the claims from a particular provider it may be a matter of concern

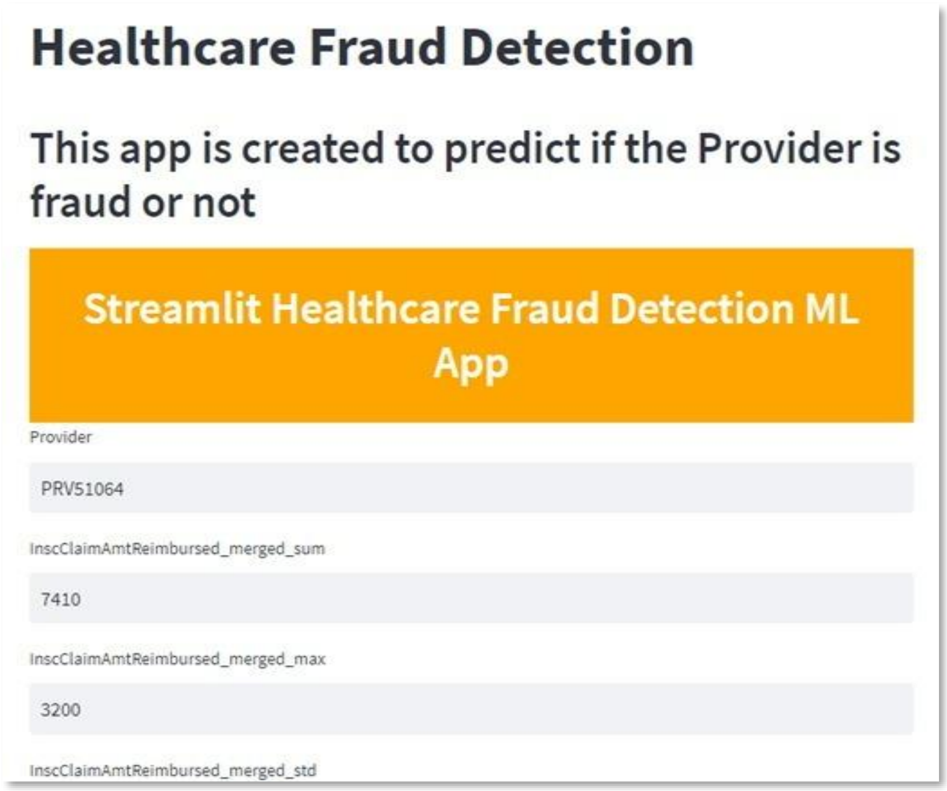
  etc.

Deployment

After building the model an interactive app was created where the insurance company can enter the details and find out if the provider is fraudulent or not.

App Link: https://healthcare-provider-fraud.herokuapp.com/

A screenshot is attached to show how the app works :



Healthcare Fraud Detection

This app is created to predict if the Provider is fraud or not

Streamlit Healthcare Fraud Detection ML App

Provider

PRV51064

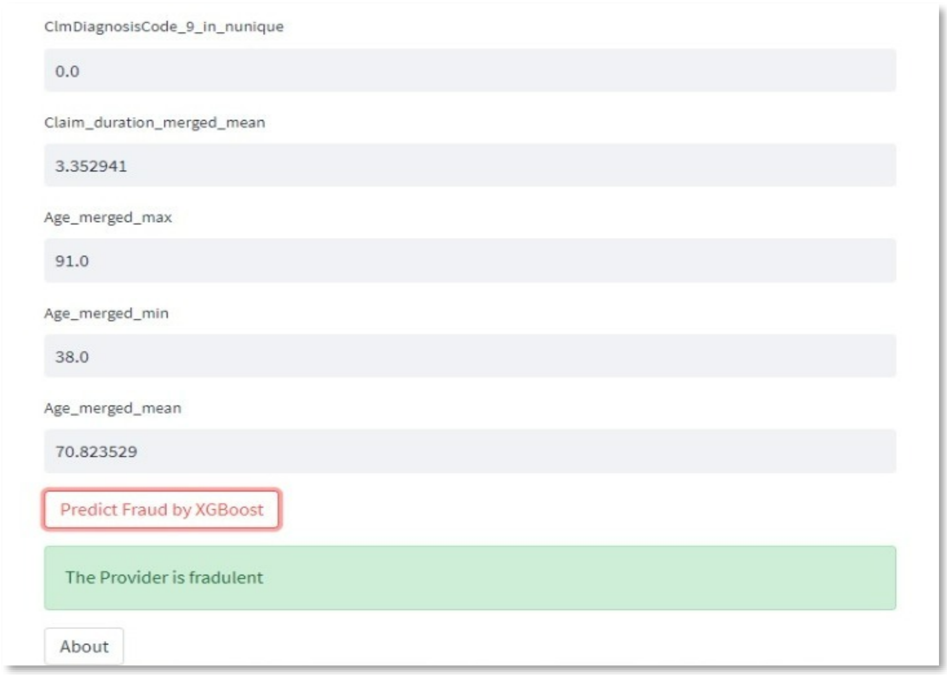
InscClaimAmtReimbursed_merged_sum

7410

InscClaimAmtReimbursed_merged_max

3200

InscClaimAmtReimbursed_merged_std



ClmDiagnosisCode_9_in_nunique

0.0

Claim_duration_merged_mean

3.352941

Age_merged_max

91.0

Age_merged_min

38.0

Age_merged_mean

70.823529

The Provider is fradulent

Conclusion

In this project the main difficulty was that it was not in appropriate format. So we had to invest quite a lot of time for data preparation. Eventually we were able to build a good model. We learnt many new things like how to make features based on insights about the data, new techniques of feature selection, fine tuning a model using bias-variance trade off, scoring a model etc.

Overall it was a great learning experience for us.