POC MAZIK

Fraud Rule Generation

**Key Dates**

23/11/2023 - Finalization of dataset

24/11/2023 - data preprocessing

28/11/2023 – Trained Model

1/12/2023 - integration with POC app

Contents

[1. Introduction 2](#_Toc151466812)

[1.1 Background 2](#_Toc151466813)

[1.2 Objectives 2](#_Toc151466814)

[2. Problem Definition 2](#_Toc151466815)

[2.1 Scope 2](#_Toc151466816)

[2.2 Data Description 2](#_Toc151466817)

[3. Data Preprocessing 2](#_Toc151466818)

[3.1 Data Cleaning 2](#_Toc151466819)

[3.2 Feature Engineering 2](#_Toc151466820)

[4. Model Training 2](#_Toc151466821)

[4.1 Model Selection 2](#_Toc151466822)

[4.2 Model Training Pipeline 2](#_Toc151466823)

[4.3 Evaluation Metrics 2](#_Toc151466824)

[5. Rule Generation 2](#_Toc151466825)

[5.1 Post-Processing 2](#_Toc151466826)

[5.2 Rule Definition 2](#_Toc151466827)

[6. Implementation 2](#_Toc151466828)

[6.1 Technology Stack 2](#_Toc151466829)

[6.2 System Architecture 2](#_Toc151466830)

[7. Testing and Validation 2](#_Toc151466831)

[7.1 Unit Testing 2](#_Toc151466832)

[7.2 Integration Testing 2](#_Toc151466833)

[7.3 Model Validation 2](#_Toc151466834)

[8. Results and Analysis 2](#_Toc151466835)

[8.1 Model Performance 2](#_Toc151466836)

[8.2 Rule Effectiveness 2](#_Toc151466837)

[9. Conclusion 2](#_Toc151466838)

[9.1 Achievements 3](#_Toc151466839)

# 1. Introduction

Fraud detection is a critical aspect of ensuring the integrity and reliability of financial systems, particularly within the healthcare domain. The proliferation of fraudulent activities poses a significant threat to the financial stability of healthcare providers and insurers. This project focuses on leveraging machine learning techniques to develop an effective fraud detection system tailored for healthcare insurance claims. By automating the detection process, the goal is to enhance the accuracy and efficiency of identifying potentially fraudulent claims, ultimately reducing financial losses and improving the overall integrity of the healthcare insurance system.

## Background

Healthcare fraud has become a pervasive issue, costing billions of dollars annually. Fraudulent activities can range from misrepresenting services to fictitious billing schemes. Traditional methods of fraud detection often fall short due to the evolving nature of fraudulent tactics. Machine learning, specifically classification algorithms, offers a promising solution by enabling systems to learn and adapt to new patterns of fraudulent behavior. In this project, we explore the application of machine learning models to analyze historical insurance claims data, identify patterns indicative of fraud, and develop a robust fraud detection model.

## 1.2 Objectives

The primary objectives of this project are:

* **Develop a Predictive Model:** Build a machine learning model capable of predicting whether a given insurance claim is likely to be fraudulent based on historical data.
* **Improve Accuracy and Efficiency**: Enhance the accuracy and efficiency of fraud detection compared to traditional rule-based methods, thereby reducing false positives and negatives.
* **Automate Fraud Detection:** Create a system that automates the fraud detection process, allowing for real-time analysis of incoming insurance claims.
* **Reduce Financial Losses:** Minimize financial losses incurred by healthcare providers and insurers due to fraudulent claims.
* **Enhance Healthcare Integrity**: Contribute to the overall integrity of the healthcare system by deterring fraudulent activities and ensuring fair and accurate claim processing.

# 2. Problem Definition

Healthcare insurance fraud poses a significant challenge in maintaining the financial integrity of healthcare systems. The problem at hand is to develop a fraud detection system using machine learning techniques to identify potentially fraudulent insurance claims. By analyzing historical data, the objective is to create a predictive model that can accurately assess the likelihood of fraud for each insurance claim. The system aims to distinguish between genuine and potentially fraudulent activities, enabling healthcare providers and insurers to take proactive measures to prevent financial losses.

## 2.1 Scope

The scope of this project encompasses the development of a predictive model that can assess the likelihood of fraud in healthcare insurance claims. The model will be trained on historical data containing information about beneficiaries, providers, medical procedures, diagnoses, and other relevant features. The predictions generated by the model will assist healthcare institutions in identifying potentially fraudulent claims for further investigation. The focus is on creating a scalable and adaptable solution that can be integrated into existing healthcare systems to automate and enhance the fraud detection process.

## 2.2 Data Description

The dataset consists of various features related to insurance claims, beneficiaries, and healthcare providers. Key data points include Beneficiary ID (BeneID), Claim ID (ClaimID), Claim Start Date (ClaimStartDt), Claim End Date (ClaimEndDt), Provider ID (Provider), reimbursement amounts, attending and operating physicians, diagnosis and procedure codes, deductible amounts, admission and discharge dates, chronic condition indicators, and potential fraud flags (PotentialFraud). The dataset provides a comprehensive view of the factors associated with each insurance claim, serving as the foundation for training the machine learning model. Understanding and processing these features are essential for developing an effective fraud detection system.

Following are the main features that are used throughout the training of the model:-

1. **BeneID**: Beneficiary ID, a unique identifier for each individual beneficiary
2. **ClaimID**: Claim ID, a unique identifier for each insurance claim.
3. **ClaimStartDt**: The start date of the insurance claim.
4. **ClaimEndDt**: The end date of the insurance claim.
5. **Provider**: Provider ID, a unique identifier for each healthcare provider.
6. **InscClaimAmtReimbursed**: The amount reimbursed by insurance for the claim.
7. **AttendingPhysician**: Physician who attended to the patient during the claim.
8. **OperatingPhysician**: Physician who performed any operations during the claim.
9. **OtherPhysician**: Other physician involved during the claim.
10. **ClmDiagnosisCode\_1 to ClmDiagnosisCode\_10**: Diagnosis codes associated with the claim. These are codes indicating the medical diagnosis.
11. **ClmProcedureCode\_1 to ClmProcedureCode\_6**: Procedure codes associated with the claim. These are codes indicating medical procedures performed.
12. **DeductibleAmtPaid**: The amount paid by the patient as a deductible.
13. **ClmAdmitDiagnosisCode**: Diagnosis code for the admission.
14. **AdmissionDt**: Date of admission.
15. **DischargeDt**: Date of discharge.
16. **DiagnosisGroupCode**: Code representing the group of diagnoses related to the claim.
17. **DOB**: Date of birth of the beneficiary.
18. **DOD**: Date of death of the beneficiary (if applicable).
19. **Gender**: Gender of the beneficiary (1 for male, 2 for female).
20. **Race**: Race of the beneficiary.
21. **RenalDiseaseIndicator**: Indicator for renal disease (Y for Yes, indicating the presence of renal disease).
22. **State**: State where the beneficiary resides.
23. **County**: County where the beneficiary resides.
24. **NoOfMonths\_PartACov**: Number of months covered by Part A of the insurance.
25. **NoOfMonths\_PartBCov:** Number of months covered by Part B of the insurance.
26. **ChronicCond\_Alzheimer to ChronicCond\_stroke:** Chronic conditions indicators (Y for Yes, N for No).
27. **IPAnnualReimbursementAmt:** Inpatient annual reimbursement amount.
28. **IPAnnualDeductibleAmt:** Inpatient annual deductible amount.
29. **OPAnnualReimbursementAmt:** Outpatient annual reimbursement amount.
30. **OPAnnualDeductibleAmt:** Outpatient annual deductible amount.
31. **PotentialFraud:** Binary indicator (0 for No, 1 for Yes) indicating potential fraud for the provider.

# 3. Data Preprocessing

Data preprocessing is a critical step in preparing the raw dataset for machine learning model training. This phase involves cleaning the data, handling missing values, and engineering new features to improve the predictive capability of the model.

## 3.1 Data Cleaning

The data cleaning process addresses issues such as missing values and inconsistencies in the dataset. The following cleaning steps have been implemented:

* **Identification and handling of duplicate physician entries:** Physicians involved in a claim (AttendingPhysician, OperatingPhysician, OtherPhysician) were analyzed to identify and address any duplications. This was achieved by creating new features (phy\_same and phy\_count) to categorize and count the involvement of physicians in each claim.
* **Calculation of claim duration:** The duration of each insurance claim was calculated by determining the difference between the ClaimStartDt and ClaimEndDt columns. The result was stored in the 'period' column.
* **Standardization of chronic condition indicators:** Columns representing chronic conditions (ChronicCond\_Alzheimer to ChronicCond\_stroke) were standardized to binary values (0 for No, 1 for Yes).
* **Handling missing dates:** Admission and discharge dates were converted to datetime objects, and missing values in the Date of Death (DOD) column were imputed using the maximum date present in the dataset.

## 3.2 Feature Engineering

Feature engineering involves creating new features that can enhance the predictive power of the model. The following feature engineering steps have been applied:

* **Creation of physician-related features:** Features like 'phy\_same' and 'phy\_count' were generated to capture the relationship between attending, operating, and other physicians in a claim. These features provide additional insights into the physician dynamics associated with each claim.
* **Calculation of patient age:** The age of the patient was computed using the Date of Birth (DOB) and Date of Death (DOD) columns. This new feature, 'age,' offers valuable information about the patient's demographic.
* **Grouping and aggregating financial features:** Financial features related to insurance claims, such as reimbursement amounts, deductible amounts, and coverage months, were grouped and aggregated at both the provider and beneficiary levels. This resulted in new columns indicating the mean and standard deviation of these financial features.

The processed dataset, containing both original and engineered features, serves as the foundation for training the fraud detection model.

# 4. Model Training

Model training is a crucial phase in the development of a fraud detection system, where the goal is to create a predictive model that accurately identifies potential instances of fraud. The chosen approach involves utilizing both a neural network implemented with TensorFlow and an Extra Trees Classifier from scikit-learn. This hybrid strategy aims to leverage the strengths of both deep learning and ensemble learning techniques.

## 4.1 Model Selection

Two distinct models have been selected for this fraud detection task:

1. **Neural Network (TensorFlow):** The neural network architecture consists of an input layer, two dense hidden layers with ReLU activation functions, and an output layer with a sigmoid activation function. The binary cross-entropy loss function is used, reflecting the nature of the binary classification task (fraudulent or non-fraudulent). The model is trained using the Adam optimizer with a learning rate of 0.0001.
2. **Extra Trees Classifier:** This ensemble learning method, implemented in scikit-learn, is chosen for its ability to handle imbalanced datasets effectively. The Extra Trees Classifier builds multiple decision trees and combines their predictions. This model is particularly useful in fraud detection scenarios where the occurrence of fraudulent cases is typically less frequent.

## 4.2 Model Training Pipeline

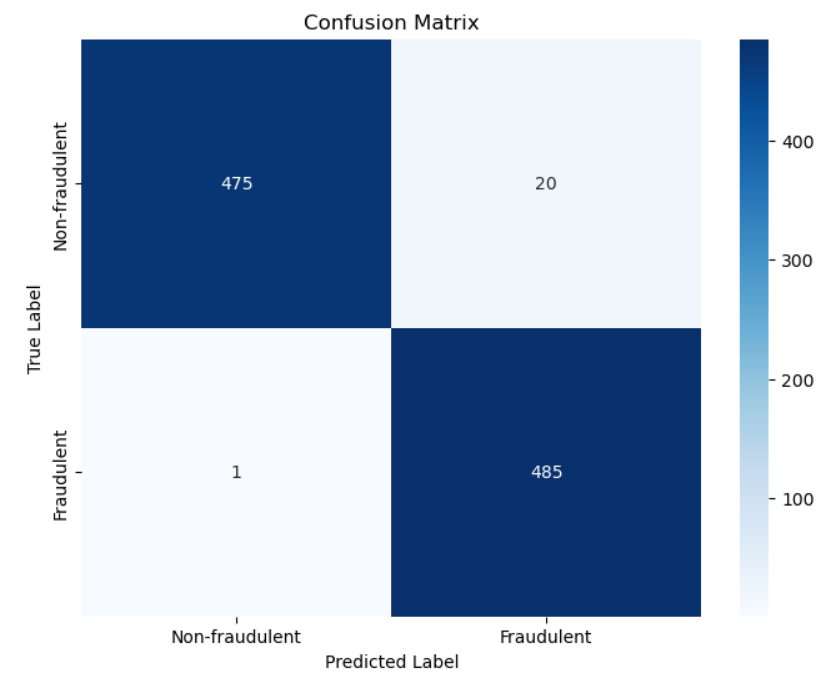
The model training pipeline involves several steps to ensure the models are trained effectively:

* **Data Resampling (SMOTE):** The dataset is resampled using Synthetic Minority Over-sampling Technique (SMOTE) to address the class imbalance issue. SMOTE generates synthetic samples for the minority class, enhancing the model's ability to recognize fraudulent cases.
* **Data Standardization:** Standardization is applied to the features using the StandardScaler from scikit-learn. This step ensures that all features have the same scale, preventing any particular feature from dominating the training process.
* **TensorFlow Neural Network Training:** The neural network is trained using the resampled and standardized data. The class weights are adjusted to handle the imbalanced nature of the dataset.
* **Extra Trees Classifier Training:** The Extra Trees Classifier is trained on the resampled and standardized data, with the training set further split into training and testing subsets.
* **Model Evaluation:** The performance of both models is evaluated using metrics such as accuracy, precision, recall, and the F1 score.

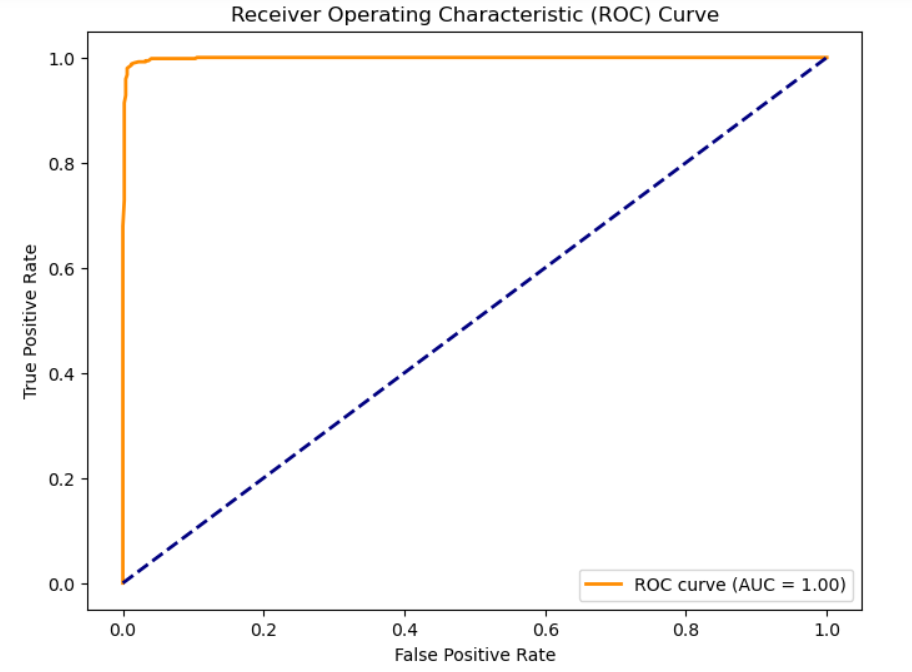
## 4.3 Evaluation Metrics

To assess the effectiveness of the models, various evaluation metrics are employed:

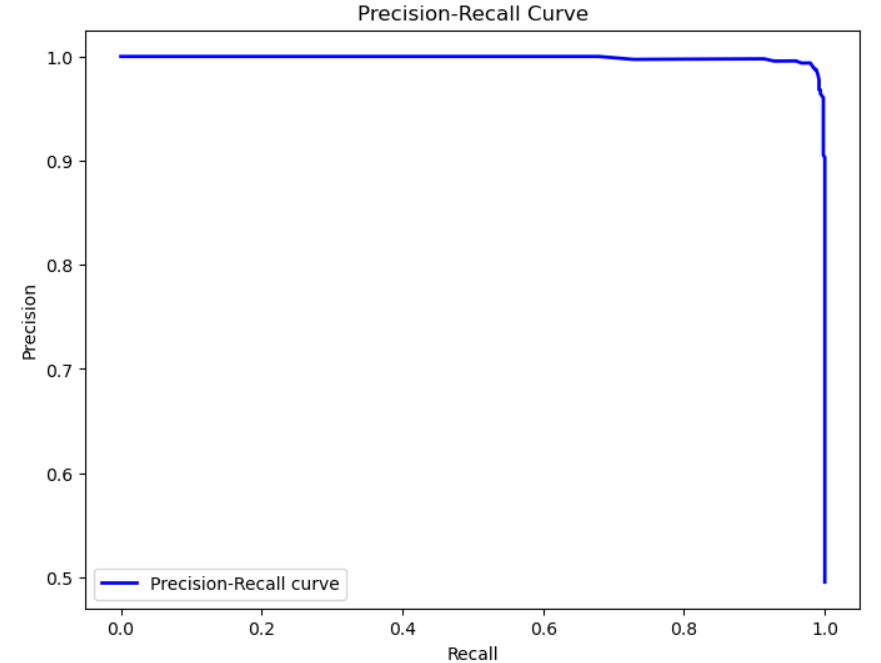
* **Confusion Matrix**: A confusion matrix is generated to visualize the true positives, true negatives, false positives, and false negatives. This matrix provides a detailed breakdown of the model's performance.
* **Classification Report:** The classification report includes metrics such as precision, recall, F1 score, and support for each class. It offers a comprehensive summary of the model's ability to correctly classify instances of fraud and non-fraud.
* **Visualization:** The confusion matrix is visualized using a heatmap, providing an intuitive representation of the model's performance in identifying fraudulent and non-fraudulent cases.



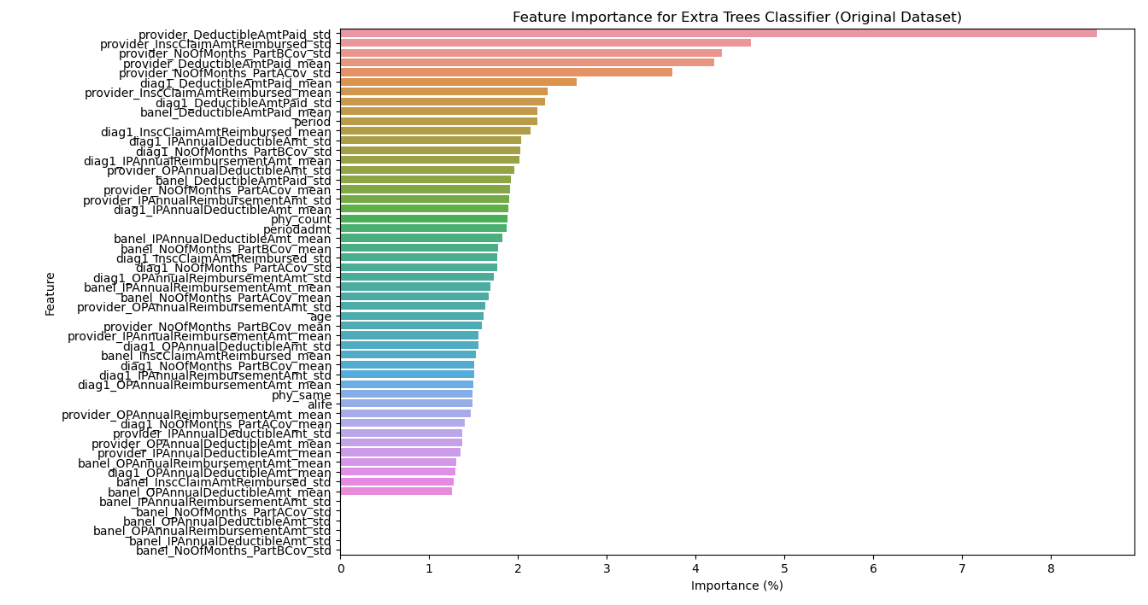
*Figure 1.01 : Confusion Matrix*

**

*Figure 1.02 : ROC Curve*

**

*Figure 1.03 : Precision-Recall Curve*

**

*Figure 1.04 : Feature Importance*

# 5. Rule Generation

## 5.1 Post-Processing

## 5.2 Rule Definition

# 6. Implementation

## 6.1 Technology Stack

## 6.2 System Architecture

# 7. Testing and Validation

## 7.1 Unit Testing

## 7.2 Integration Testing

## 7.3 Model Validation

# 8. Results and Analysis

## 8.1 Model Performance

## 8.2 Rule Effectiveness

# 9. Conclusion

## 9.1 Achievements