Machine Learning Project

Healthcare Provider Fraud Detection

Project Report

Academic Year:

2024-2025

Submitted By:

Paras Bramhankar Div-2, 37
Komal Buche Div-2, 28

1.Introduction

- **Healthcare Fraud:** White-collar crime involving false claims for financial gain.
- **Impact:** Affects healthcare funds and individual members.
- **Provider Fraud:** Misuse of the system by service providers, especially in Medicare.
- **Organized Crime:** Often involves healthcare providers, physicians, and beneficiaries conspiring to inflate costs.
- **Tactics:**
    - Use of ambiguous diagnosis codes.
    - Submission of fraudulent claims.
- **Consequences:**
    - Increased insurance premiums.
    - Higher healthcare costs.
- **Common Fraud Practices:**
    - Billing for services not provided.
    - Duplicate claims.
    - Misrepresentation of services.
    - Charging for more expensive procedures than delivered.
- **Importance of Addressing:** Ensures the integrity and affordability of healthcare systems.

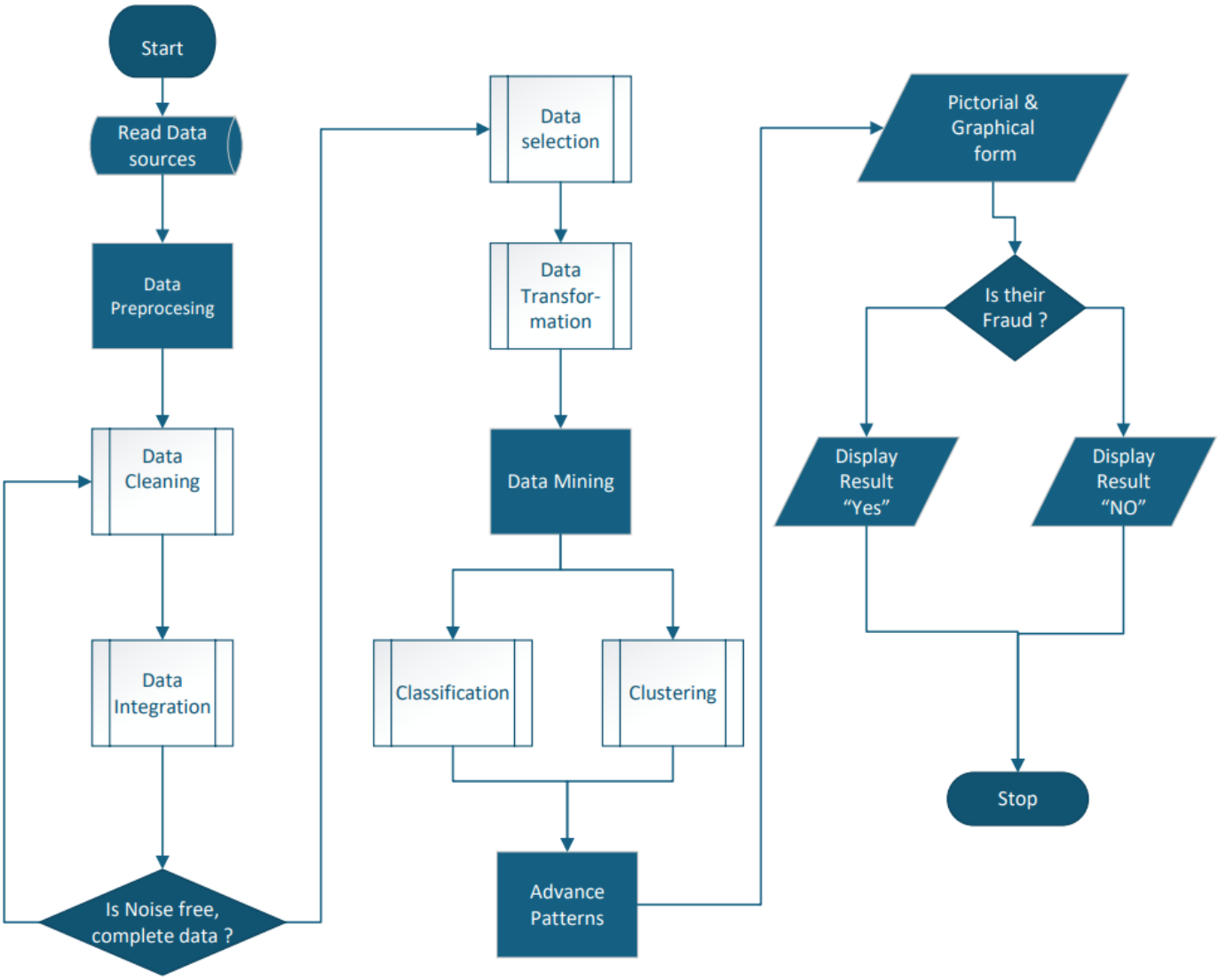
2. Problem Statement

- Healthcare fraud involves the submission of false or misleading claims for financial gain.
- It affects various stakeholders, including healthcare funds, insurance companies, and individual members.
- Provider fraud, such as misrepresentation and billing for unprovided services, is a significant contributor to Medicare misuse.
- Organized schemes often involve collaboration between providers, physicians, and beneficiaries to exploit the system.
- Fraudulent practices lead to increased Medicare spending and insurance premiums, making healthcare unaffordable for many.
- Inefficient detection mechanisms allow such practices to persist, necessitating improved solutions to combat fraud and protect healthcare systems.

Technologies and Tools:

- **Programming Language:** Python
- **IDE:** Visual Studio Code, Jupyter Notebook
- **Libraries:** Pandas, NumPy, Scikit-learn , Matplotlib/Seaborn (for visualization)
- **Data Source**: Kaggle datasets or other healthcare databases
- **Model Deployment**: Flask for web services, Heroku for cloud deployment

3.Flowchart



4. Dataset Description

The dataset used in this project was sourced from Kaggle and focuses on healthcare provider fraud detection. It consists of eight CSV files divided into training and testing data for four categories:

1. **Beneficiary Dataset (train/test)**
   This dataset includes KYC (Know Your Customer) details of beneficiaries such as demographic information, health conditions, and regional data.
   - **Key Features**:
     - BeneID: Unique identifier for the patient.
     - DOB: Patient's date of birth.
     - DOD: Date of death (if applicable).
     - Gender, Race, State, Country: Demographic details of the patient.
     - RenalDiseaseIndicator: Indicates if the patient has renal disease (Yes/No).
     - CronicCondi_*: Flags indicating chronic conditions like Alzheimer's, heart failure, diabetes, etc.
     - IPAnnualReimbursementAmt: Total annual reimbursement for inpatient claims.
     - IPAnnualDeductibleAmt: Annual deductible amount paid for inpatient care.
     - OPAnnualReimbursementAmt: Total annual reimbursement for outpatient claims.
     - OPAnnualDeductibleAmt: Annual deductible amount paid for outpatient care.
2. **Inpatient Dataset (train/test)**
   This dataset contains details about claims filed for patients admitted to hospitals.
   - **Key Features**:
     - BeneID: Unique identifier for the patient.
     - ClaimID: Unique identifier for each claim.
     - ClaimStartDt and ClaimEndDt: Dates indicating the start and end of the claim period.
     - Provider: Unique identifier for the healthcare provider.

        - InscClaimAmtReimbursed: Amount reimbursed for the claim.
        - AttendingPhysician, OperatingPhysician, OtherPhysician: Unique IDs for attending, operating, and other physicians involved.
        - DeductibleAmtPaid: Total deductible paid by the patient.
        - ClmAdmitDiagnosisCode: Code for the diagnosis requiring hospitalization.
        - ClmDiagnosisCode_(1–10): Diagnosis codes indicating patient conditions (up to 10).
        - ClmProcedureCode_(1–6): Codes for procedures performed on the patient (up to 6).
        - AdmissionDate and DischargeDate: Dates of hospital admission and discharge.
        - DiagnosisGroupCode: Group code for the diagnosis performed on the patient.

3. **Outpatient Dataset (train/test)**
   This dataset contains details of claims filed for patients who visited the hospital but were not admitted.
    - **Key Features**:
      Similar to the inpatient dataset, excluding AdmissionDate, DischargeDate, and DiagnosisGroupCode.

4. **Label Dataset (train/test)**
   This dataset provides class labels for providers to indicate whether they are fraudulent or not.
    - **Key Features**:
        - Provider: Unique identifier for the healthcare provider.
        - Class: Indicates fraud status (1 for fraud, 0 for non-fraud).

Summary of Dataset Usage

- **Training Data**: Includes all relevant details with class labels for providers.
- **Testing Data**: Includes details for providers without class labels (used for prediction).

5. Methodology

The dataset enables the identification of patterns in claims, helping to build models for detecting healthcare fraud efficiently.

The project follows a systematic methodology to detect and analyze healthcare fraud, comprising the following steps:

1. **Data Collection and Exploration**
    - The dataset is sourced from Kaggle and consists of multiple CSV files: beneficiary data, inpatient data, outpatient data, and labels for fraud classification.
    - Explore the datasets to understand key attributes, relationships, and distribution of fraud vs. non-fraud cases.
    - Summarize and visualize the data for insights (e.g., claim amount trends, provider activities, and diagnosis patterns).
2. **Data Preprocessing**
    - **Handle Missing Values**: Replace or remove missing data in important attributes like diagnosis codes and reimbursement amounts.
    - **Data Cleaning**: Eliminate duplicates and inconsistencies across the dataset.
    - **Feature Engineering**:
        - Create new features such as claim frequency, average reimbursement per provider, and high-risk diagnosis combinations.
        - Scale numerical features like claim amounts and deductible amounts for better model performance.
3. **Frequent Pattern Mining**
    - Apply **Apriori Algorithm** to identify frequently occurring combinations of diagnosis codes, procedures, and reimbursement patterns.
    - Use these patterns to generate rules that can flag potential fraud scenarios.
4. **Data Splitting**
    - Divide the dataset into training and testing subsets for building and evaluating the classification model.
5. **Classification**
    - Train a **Random Forest Classifier** to classify healthcare providers or claims as fraudulent or non-fraudulent.
    - Use key features from the dataset, such as reimbursement amounts, diagnosis codes, procedure codes, and provider details, as input to the model.
6. **Model Evaluation**
    - Evaluate the classifier's performance using metrics like:
        - **Accuracy**: Overall correctness of the model.
        - **Precision and Recall**: To measure how effectively fraudulent cases are detected.

        - **F1-Score**: A balance between precision and recall.
    - Visualize the results using confusion matrices and performance graphs for better insights.
7. **Insights and Conclusion**
    - Analyze the results to identify common traits of fraudulent claims.
    - Summarize findings and propose actionable recommendations for stakeholders to mitigate fraud.
8. **Future Work**
    - Propose further enhancements like real-time fraud detection, incorporating more advanced algorithms, or using external datasets for validation.

5.Implementation Steps

1. Problem Definition and Requirements Gathering

- Define the scope of fraud detection (e.g., overbilling, billing for services not rendered).
- Gather requirements from stakeholders, including legal and regulatory considerations.
- Also define python environment like conda, venv.

Step 1: Setting Up a Virtual Environment (venv)

1. **Install** venv :
    - Ensure you have venv installed. If not, install it using:
      pip install virtualenv
2. **Create a Virtual Environment**:
    - Create a new virtual environment:
      python -m venv healthcare_fraud
3. **Activate the Environment**:
    - **On Windows:**
      bash
      .\healthcare_fraud\Scripts\activate
    - **On macOS/Linux:**
      bash
      source healthcare_fraud/bin/activate

Step 2: Installing Required Packages

- **Once your environment is set up, install the necessary libraries for your project:**

  pip install pandas numpy scikit-learn matplotlib seaborn jupyter

2. Data Collection and Integration

- Collect data from various sources such as medical claims, patient records, provider information, and billing data.

- Integrate data from multiple sources into a cohesive dataset.
- **Library:** pandas, numpy, sqlalchemy, pymongo
- **read_csv() – pandas**: Load data from CSV files
- **read_sql() – pandas**: Load data from a SQL database
- **connect()** – sqlalchemy: Connect to a database
- **find()** – pymongo: Retrieve data from MongoDB

3. Data Preprocessing

- **Data Cleaning**: Remove duplicates, correct errors, and handle missing values.
- **drop_duplicates() – pandas**: Remove duplicates
- **fillna() – pandas**: Handle missing values
- replace() – pandas: Correct errors
- **Data Transformation**: Normalize and standardize data, convert categorical variables into numerical ones if needed.
- **StandardScaler() – sklearn.preprocessing**: Standardize data
- **MinMaxScaler() – sklearn.preprocessing**: Normalize data
- **LabelEncoder() – sklearn.preprocessing**: Encode categorical variables

4. Exploratory Data Analysis (EDA)

- Visualize the data to understand distributions and identify patterns.
- Use techniques like histograms, bar charts, and box plots to summarize data characteristics.
- **Library:** matplotlib, seaborn, pandas-profiling
- **hist()** – matplotlib.pyplot: Plot histograms
- **boxplot()** – seaborn: Create box plots
- **barplot()** – seaborn: Create bar charts
- **profile_report()** – pandas-profiling: Generate an EDA report

5. Feature Engineering

- Identify and create relevant features that could indicate fraudulent activity (e.g., average billing amount, frequency of claims).
- Use domain knowledge to generate meaningful features.
- **Library:** pandas, numpy, sklearn.feature_extraction
- **apply() – pandas**: Create new features

- **groupby() – pandas**: Aggregate data
- **PolynomialFeatures()** – sklearn.preprocessing: Generate polynomial features
- **FeatureHasher() – sklearn.feature_extraction**: Convert categorical data to features

6. Model Building

- **Supervised Learning**: Use labeled data to train classification models like Logistic Regression, Decision Trees, Random Forests, or Gradient Boosting.
- **LogisticRegression() – sklearn.linear_model**: Logistic Regression
- **DecisionTreeClassifier() – sklearn.tree**: Decision Tree
- **RandomForestClassifier() – sklearn.ensemble**: Random Forest
- **XGBClassifier() – xgboost**: Gradient Boosting
- **Unsupervised Learning**: Use clustering algorithms like K-means or DBSCAN to identify anomalies in the data.
- **KMeans() – sklearn.cluster**: K-means clustering
- **DBSCAN() – sklearn.cluster**: DBSCAN

7. Model Evaluation and Selection

- Evaluate models using metrics like accuracy, precision, recall, F1-score, and ROC-AUC.
- Select the model that performs best and meets the project requirements.
- **Library:** sklearn.metrics, matplotlib
- **accuracy_score() – sklearn.metrics**: Calculate accuracy
- **precision_score() – sklearn.metrics**: Calculate precision
- **recall_score() – sklearn.metrics**: Calculate recall
- **f1_score() – sklearn.metrics**: Calculate F1-score
- **roc_auc_score() – sklearn.metrics**: Calculate ROC-AUC
- **plot_roc_curve() – matplotlib**: Plot ROC curve

8. Deployment

- Implement the chosen model into a production environment.
- Set up a monitoring system to track model performance and retrain it periodically with new data.
- **Library**: flask, fastapi, joblib, mlflow
- **dump() – joblib**: Save the model
- **load() – joblib**: Load the model

- **run() – flask**: Run Flask app
- **serve() – fastapi**: Serve FastAPI app
- **log_model() – mlflow**: Log and deploy the model

9. Interpretability and Reporting

- Use techniques like SHAP (SHapley Additive exPlanations) to explain model predictions.
- Generate reports and dashboards for stakeholders to understand the findings and make informed decisions.
- **Library**: shap, lime, dash, plotly
- **Explainer() – shap:** SHAP explainability
- **explain_instance() – lime.lime_tabular**: LIME explainability
- **dash.Dash() – dash:** Create dashboards
- **plot() – plotly.graph_objects**: Generate interactive plots

10. Continuous Improvement

- Regularly update the system with new data and feedback from users.
- Continuously improve the model and its performance based on real-world results and changing patterns in fraud behavior.
- **Library:** mlflow, airflow, prefect
- **start_run() – mlflow:** Track experiments
- **dag() – airflow:** Define Airflow DAG for automation
- **flow() – prefect**: Automate workflows

5. Future Work

- We can Extend this work by solving this problem with deep learning and experimenting with different numbers of layers, dropout rate, filter size and max-pooling layer to get a better model.
- Regularly update the system with new data and feedback from users.
- Continuously improve the model and its performance based on real-world results and changing patterns in fraud behavior.

References:

- **Python Documentation:** https://docs.python.org/3/
- **DataSets:** https://www.kaggle.com/datasets/rohitrox/healthcare-provider-fraud-detection-analysis

- **Github:** https://github.com/
- **Jupyter Notebook Documentation:** https://docs.jupyter.org/en/latest/
- **Flask's Documentation:** https://flask.palletsprojects.com/en/stable/
- **MDN reference:** https://developer.mozilla.org/en-US/curriculum/
- **Heroku deployment:** https://www.heroku.com/