GST Fraud Detection Using Machine Learning – Read me file

Project Overview

The GST Fraud Detection project is a smart analytics model designed to help tax authorities detect and prevent fraudulent activities within India's GST system. By leveraging machine learning, the platform can analyse large datasets of business transactions, detect suspicious activities, and provide real-time insights for decision-makers. It not only detects anomalies but also predicts future trends and sectors with a high risk of tax evasion.

Features

- **Fraud Detection**: Identify fraudulent GST activities, such as fake invoicing and non-compliant businesses.
- **Revenue Prediction**: Predict future revenue trends and identify industries or businesses with a higher risk of tax evasion.
- **Machine Learning Models**: Employ Random Forest models to ensure high accuracy and performance in fraud detection.

Technologies Used

- **Programming Language**: Python
- **Machine Learning Libraries**:
  - scikit-learn (Random Forest, SMOTE, metrics)
  - pandas, numpy (Data manipulation and analysis)
  - matplotlib, seaborn (Data visualization)
- **Data Preprocessing**: Imputation for missing data, outlier removal, feature scaling, and dataset balancing (SMOTE)

Installation

To set up and run the project locally, follow these steps:

1. **Clone the Repository**:

bash

Copy code

git clone https://github.com/yourusername/gst-fraud-detection.git cd gst-fraud-detection

2. **Set Up a Virtual Environment**: It's recommended to use a virtual environment to manage dependencies.

bash

Copy code

```
python3 -m venv venv
source venv/bin/activate   # On Windows: venv\Scripts\activate
```

3. **Install Required Dependencies**: Install the necessary Python libraries from the requirements.txt file.

bash

Copy code

```
pip install -r requirements.txt
```

Data Preparation

The dataset used in this project consists of business transaction records, tax filings, and compliance data. Before training the machine learning models, several preprocessing steps were applied:

1. **Handling Missing Data**: Missing values were imputed using median or mode imputation, depending on the data type.
2. **Outlier Removal**: Outliers in financial features were removed using the interquartile range method.
3. **Class Balancing**: The dataset was imbalanced, with far more non-fraudulent cases. We used **SMOTE** to balance the classes and ensure the model could identify fraudulent cases effectively.
4. **Feature Scaling**: The data was scaled using techniques like Min-Max Scaling and Standard Scaling to normalize the data and ensure consistent model performance.

Dataset Structure:

- **X_Train_Data_Input.csv**: Features of the training dataset.
- **Y_Train_Data_Target.csv**: Labels (fraud or not fraud) for the training dataset.
- **X_Test_Data_Input.csv**: Features of the testing dataset.
- **Y_Test_Data_Target.csv**: Labels for the testing dataset.

Usage

Running the Model: Once the data is prepared, you can run the machine learning model to detect fraud.

- **Train the Model**: The model will be trained on the preprocessed data.
- **Test the Model**: After training, the model is tested on the test dataset to evaluate its performance.

Results and Evaluation

The machine learning model, specifically the Random Forest Classifier, demonstrated strong performance:

- **Accuracy**: 96%
- **Precision**: 97%
- **Recall**: 96%
- **ROC-AUC Score**: 0.98

These metrics indicate that the model is highly effective at detecting fraudulent activity while minimizing false positives and false negatives. The attached visualizations, such as the confusion matrix and ROC-AUC curve, provide further insights into the model's performance.

Steps to Follow Before Evaluating the Model

To ensure the process runs smoothly and consistently for model evaluation, the following steps need to be executed:

Data Preparation

Before the model can be evaluated on the evaluation dataset, it is important to ensure that the same preprocessing steps used on the training and test datasets are applied. Here are the specific steps for preparing the data:

Handling Missing Data

- Check for missing values in the evaluation dataset (similar to how missing values were handled in the training and test datasets).
    - For columns with missing values:
        - Numerical features: Apply median imputation to fill missing values, as this technique was used to maintain the robustness of the data without being affected by outliers.

        - Categorical or low-variance features: Use mode imputation to fill missing values.

This will ensure that the model can work effectively without being interrupted by missing data during the evaluation phase.

Outlier Detection and Removal

- Similar to the training and test datasets, the evaluation dataset might contain outliers that could distort the model's predictions.
    - Apply the Interquartile Range (IQR) method to detect and remove outliers. This step ensures that the model's performance is not biased by extreme, unrepresentative data points.

Class Imbalance Handling

- If the evaluation dataset has a significant class imbalance (more non-fraudulent cases than fraudulent ones), the model's performance may be affected.
    - Consider using SMOTE if the class distribution in the evaluation dataset is skewed. This will generate synthetic cases for the minority class (fraudulent cases) to ensure balanced performance.