**Insurance Claims Model**

**Project Overview**

This project aims to build a machine learning model to predict two target variables, "Coverage Code" and "Accident Source," from a dataset containing insurance claim descriptions. The model handles highly imbalanced data and uses Natural Language Processing (NLP) techniques to transform the text data into embeddings, which are then used to train a Random Forest classifier. The project includes a graphical user interface (GUI) to allow users to upload their dataset, run the model, and view the evaluation results.

**Folder Structure**

The folder structure of the project is as follows:

insurance-claims-model/

├── input/ # Input folder for raw data

├── output/ # Output folder for results (reports, models, embeddings)

│ ├── models/ # Folder to store trained models

│ ├── embeddings/ # Folder to store generated embeddings

│ ├── reports/ # Folder to store evaluation reports and logs

├── src/ # Source code for model and utilities

│ ├── \_\_init\_\_.py

│ ├── train\_model.py # Script to train and evaluate the model

│ ├── utils.py # Helper functions for data processing and model handling

├── gui/ # GUI-related files

│ ├── insurance\_gui.py # GUI implementation for dataset upload and model execution

├── .gitignore # Git ignore file for excluded files/folders

├── requirements.txt # List of Python dependencies for the project

├── README.md # Project documentation

├── output\_folder\_structure.txt # Output folder structure details (optional)

**Installation**

**Dependencies**

This project requires several Python libraries to function properly. You can install the required dependencies using the requirements.txt file.

1. Clone the repository:

git clone https://github.com/yourusername/insurance-claims-model.git

cd insurance-claims-model

1. Install the dependencies:

pip install -r requirements.txt

This will install all necessary libraries including pandas, numpy, scikit-learn, sentence-transformers, imblearn, and others.

**Usage**

**Running the Model with GUI**

1. Launch the GUI:

python gui/insurance\_gui.py

1. The GUI will open with the following options:
   * **Upload Dataset**: Choose the input Excel file containing the insurance claim data.
   * **Run**: Execute the model with the provided dataset.
2. After execution, the GUI will display:
   * **Evaluation Results**: Precision, Recall, F1-score, and ROC AUC for the predicted targets.
   * **Save Results**: The evaluation results will be saved as an Excel file in the output/reports/ folder.
3. The trained model and embeddings are saved in the output/models/ and output/embeddings/ folders, respectively.

**Model Explanation**

The project uses the following approach:

**Data Preprocessing**

* The dataset contains two target variables: "Coverage Code" and "Accident Source."
* These target variables are highly imbalanced, with some classes having only a few instances.
* **Embeddings**: The Claim Description text is converted into embeddings using the Sentence-Transformers library. These embeddings represent the text in a dense vector space, allowing the model to capture the semantic meaning of the descriptions.

**Handling Imbalance**

* **SMOTE (Synthetic Minority Over-sampling Technique)** is applied to handle the class imbalance for the target variables. It generates synthetic samples for the minority classes to balance the dataset.
* **Class Weights**: In addition to SMOTE, the Random Forest classifier uses class weights to give higher importance to the minority classes during training.

**Modeling**

* A **Random Forest Classifier** is used to train the model, as it is effective for handling high-dimensional data and imbalanced classes.
* Hyperparameter tuning is done using **RandomizedSearchCV** to find the optimal parameters for the classifier.

**Evaluation Metrics**

* The model's performance is evaluated using multiple metrics:
  + **Precision**
  + **Recall**
  + **F1-Score**
  + **ROC AUC** (for multi-class classification)

**Output**

**Folders and Files:**

1. **Models**: The trained Random Forest model is saved in the output/models/ folder.
   * File: optimized\_random\_forest\_model.pkl
2. **Embeddings**: The embeddings for the Claim Description are saved in the output/embeddings/ folder.
   * File: X\_train\_embeddings.pkl
3. **Reports**: The evaluation results (Precision, Recall, F1-Score, ROC AUC) are saved in the output/reports/ folder as an Excel file.
   * File: model\_evaluation\_results.xlsx
4. **Logs**: Any logs related to the training process, errors, or warnings can be saved in this folder (optional).

**Important Files**

* train\_model.py: Contains the code for training, hyperparameter tuning, and model evaluation.
* utils.py: Helper functions for data preprocessing, embedding extraction, and model handling.
* insurance\_gui.py: The GUI interface that allows users to upload datasets, run the model, and view results.