Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

To develop a predictive machine learning model that automates the identification of fraudulent claims in auto insurance, thereby enhancing claims management and reducing manual investigation efforts.

1.2 Scope

This system will:

- Analyze past insurance claim data.

- Identify patterns of fraudulent behavior.

- Predict whether a new claim is likely to be fraudulent.

- Provide explainability for decisions (optional advanced feature).

1.3 Stakeholders

- Insurance companies and their fraud investigation units.

- Data scientists and engineers.

- End-users (claim managers, auditors).

2. Functional Requirements

FR1: Data Ingestion - Load and preprocess insurance claims data.

FR2: Feature Engineering - Derive meaningful features like claim amount ratio, customer behavior, etc.

FR3: Model Training - Train classification models (e.g., Logistic Regression, Random Forest, XGBoost).

FR4: Model Evaluation - Evaluate model performance using metrics like Accuracy, Precision, Recall, F1-Score, AUC.

FR5: Fraud Prediction - Predict whether an incoming claim is fraudulent.

FR6: Model Explainability (Optional) - Use SHAP/LIME to interpret predictions.

FR7: Visualization Dashboard (Optional) - Build a simple web interface to view predictions and stats.

3. Non-Functional Requirements

- Scalability: Should handle large claim datasets.
- Performance: Model prediction latency should be low.
- Usability: Simple CLI or web-based dashboard.
- Maintainability: Modular code design for easy updates.
- Security: Mask or anonymize PII if deployed online.

4. Data Description

Based on your uploaded CSV file, the dataset likely contains features such as:

- claim_amount
- policy_number
- insured_age
- vehicle_type
- accident_type
- claim_reason
- fraud_reported (target column)

5. Proposed ML Pipeline

1. Data Collection (.csv)
2. Data Cleaning & Preprocessing
3. Exploratory Data Analysis (EDA)
4. Feature Engineering
5. Model Selection (Logistic, RandomForest, XGBoost)
6. Model Evaluation (Accuracy, AUC, etc.)
7. Model Tuning

8. Fraud Detection Predictions

9. Deployment (Optional: Streamlit/FastAPI)

6. Tech Stack

Language: Python 3.x

ML Libraries: scikit-learn, XGBoost

Visualization: Matplotlib, Seaborn

Dashboard: Streamlit / React

Deployment: Flask / FastAPI (optional)

Data Storage: CSV (for now)

7. Milestones

Phase 1: Data Understanding & Cleaning - 1 day

Phase 2: EDA + Feature Engineering - 1-2 days

Phase 3: Modeling & Evaluation - 2-3 days

Phase 4: Explainability + Tuning - 1-2 days

Phase 5: Deployment (Optional) - 1-2 days