Machine learning Engine

The decision to transition from traditional rule-based methods (if-else conditions) to machine learning for fraud detection depends on various factors related to the size and complexity of the data, as well as the nature of the patterns present in the data. Criteria indicating the need for a shift to Machine learning includes the presence of complex fraud patterns that are difficult to express with simple rules. Moreover, if dealing with large, diverse datasets containing a substantial amount of transaction data, machine learning can effectively handle the complexity and scale, outperforming rule-based systems. Additionally, the adaptability to changing fraud patterns over time favors machine learning models, which can continuously learn from new data and provide a more flexible solution compared to static rule-based systems.

Objective: Develop a robust machine learning solution for fraud detection in financial transactions using a combination of RandomForestClassifier and XGBoostClassifier models, and ect.

High-Level Requirements:

Functional:

- The system should detect fraudulent transactions in financial data with high accuracy (e.g., >85% accuracy).
- The system should provide real-time fraud alerts for flagged transactions.
- The system should allow training and deployment of multiple machine learning models.
- The system should track and report on the performance of deployed models.

Non-Functional:

- The system should be able to handle large volumes of financial transactions with minimal latency.
- The system should be robust against data and model manipulation attempts.
- The system should be secure and compliant with relevant data privacy regulations.
- The system should be user-friendly for administrators and analysts.

Low-Level Requirements:

Data Loading and Preprocessing:

- The system should support loading data from various formats like CSV, JSON, and relational databases.
- The system should be able to handle missing and invalid data values.
- The system should implement feature engineering techniques like time-based features and one-hot encoding.
- The system should allow the configuration of specific features and transformation methods.

Model Training:

- The system should support the training of RandomForestClassifier and XGBoostClassifier models.
- The system should allow setting hyperparameters for each model like a number of estimators and learning rate.
- The system should enable splitting data into training, validation, and test sets with configurable ratios.
- The system should track training metrics like accuracy, precision, recall, and F1-score.

Model Saving and Deployment:

- The system should save trained models in a readily deployable format (e.g., pickle, joblib).
- The system should provide an interface for deploying models to a production environment.
- The system should facilitate versioning and switching between different trained models.

Prediction and Evaluation:

- The system should allow real-time or batch prediction on new financial transactions.
- The system should display prediction results with confidence scores for each model.
- The system should calculate and report relevant evaluation metrics like accuracy, precision, recall, and F1-score on the test set.

Results and Status Reporting:

- The system should display the accuracy of both models for stakeholders to assess performance.
- The system should provide real-time or periodic reports on model performance and fraud detection statistics.
- The system should integrate with existing reporting systems or APIs for wider dissemination of information.