Anomaly Detection for Financely- Report

1. Introduction to Isolation Forest

Isolation Forest is an unsupervised machine learning algorithm primarily used for anomaly detection. Unlike other methods that profile normal data points, Isolation Forest works on the principle of isolating anomalies. Its key characteristics include:

- **Isolation-Based Anomaly Detection:** Anomalies are few and different. The algorithm randomly partitions data points by selecting features and split values until each point is isolated. Points that require fewer splits to isolate are more likely to be anomalies.
- **Efficiency and Scalability:** Isolation Forest is computationally efficient, particularly with large datasets, as it constructs relatively shallow trees and works in linear time complexity.
- **No Need for Prior Labeling:** Being an unsupervised method, it doesn't require pre-labeled data for training, making it suitable for scenarios where anomalous examples are rare or not known in advance.

---

2. How Isolation Forest is Trained

2.1. Data Preparation

Before training the Isolation Forest model, data must be properly prepared and preprocessed. In **Financely**, historical transaction data is fetched via an API, and the following steps are taken:

- **Data Collection:** Transactions are pulled from the backend using an API endpoint.
- **Handling Missing Values:** Any rows with invalid or missing dates (e.g., "NaN-NaN-NaN") are removed to ensure data quality.
- **Date Parsing:** The date string is converted into a Python datetime object, which then allows extraction of temporal features.

2.2. Feature Engineering

Key features are derived from the transaction data to capture temporal patterns and categorical information. The following transformations are applied:

- **Temporal Features:**
    - **Hour:** The hour component of the transaction time.
    - **Day of Week:** Extracted from the date to determine the weekday.

    - **Time of Month:** The day of the month when the transaction occurred.
    - **Weekend Indicator:** A binary flag (1 or 0) indicating whether the transaction happened on a weekend.
- **Amount Scaling:**
    - **Standardization:** The amount field is scaled using StandardScaler to normalize the feature distribution, ensuring that large numerical differences do not skew the model.
- **Categorical Encoding:**
    - **One-Hot Encoding:** The transaction category (e.g., "Food", "Rent", "Shopping", etc.) is transformed using OneHotEncoder to convert categorical data into numerical columns. Each unique category becomes a binary feature.

2.3. Model Training

With the engineered features ready, the Isolation Forest model is trained using the following steps:

- **Feature Selection:** The final feature set includes the scaled amount, hour, day of week, time of month, weekend flag, and one-hot encoded category columns.
- **Hyperparameters:** Based on experimental evaluation (as noted in a separate notebook, isolation_forest_evaluation.ipynb), the following hyperparameters were chosen for optimal performance:
    - **n_estimators:** 100 (the number of trees in the forest)
    - **contamination:** 0.05 (expected proportion of outliers in the dataset)
    - **random_state:** 42 (to ensure reproducibility)
- **Model Fitting:** The Isolation Forest model is fitted to the prepared data. During training, the model learns to isolate data points in the feature space. Points that are easily isolated (i.e., requiring fewer splits) are considered anomalies.
- **Serialization:** After training, the model, along with the StandardScaler and OneHotEncoder, is saved using Python's pickle module. This allows the trained objects to be reloaded for real-time prediction during inference.

---

3. Implementation in Financely

3.1. Training Pipeline Integration

Within the **Financely** project, the model training pipeline is integrated into the data processing module. The pipeline includes:

- **Data Fetching:** A function (fetch_transactions()) makes an API call to retrieve historical transaction data from the backend.
- **Preprocessing & Feature Engineering:** As described earlier, the raw data undergoes transformation into a format suitable for the Isolation Forest model.
- **Model Training:** The Isolation Forest model is trained on the selected features. Experimentation led to the adoption of specific hyperparameters to best suit the characteristics of the financial transaction data.
- **Model Storage:** The trained model and preprocessing tools (scaler and encoder) are stored to disk. This ensures that the same transformations can be applied during inference, maintaining consistency between training and evaluation.

3.2. Backend Deployment

The trained Isolation Forest model is deployed as part of a Flask-based backend service. The key elements of this deployment are:

- **Loading Trained Objects:** Upon starting, the Flask server loads the pre-trained Isolation Forest model, along with the scaler and encoder, from their serialized files.
- **Real-Time Feature Engineering:** A dedicated function (feature_engineering()) applies the same preprocessing steps to incoming transaction data as were used during training. This ensures the model receives data in the expected format.
- **Anomaly Detection API:**
    - The API endpoint (/detect_anomaly) accepts POST requests containing new transaction data.
    - The endpoint processes the input through feature engineering, then passes the resulting feature vector to the Isolation Forest model.
    - Based on the model's prediction, the endpoint returns a JSON response indicating whether the transaction is "Normal" or an "Anomaly."

3.3. Frontend Integration

In the user-facing component of **Financely**, the anomaly detection system is tightly integrated into the transaction submission workflow:

- **User Interaction:** When a user attempts to add a transaction, the frontend submits the transaction details to the anomaly detection API.
- **Alert Mechanism:** If the backend returns an "Anomaly" prediction, the frontend triggers a prompt (using a modal or alert, e.g., via Swal.fire) to warn the user that the transaction appears unusual. The user is then given the option to either proceed or cancel the transaction.

- **Decision Handling:** The user's response (i.e., to ignore or acknowledge the anomaly) dictates whether the transaction is finally submitted or halted, adding an extra layer of verification against potential errors or fraud.

---

4. Evaluation of the Model

- This model achieves an **accuracy of 94.63%.**

4.1. Evaluation Methodology

Evaluation of the Isolation Forest model in **Financely** involves several aspects:

- **Hyperparameter Tuning:** Initial experimentation (documented in a separate evaluation notebook named isolation_forest_evaluation.ipynb) involved tuning hyperparameters such as n_estimators and contamination to achieve the best balance between detecting true anomalies and minimizing false positives.
- **Accuracy Score:** To get the accuracy score, synthetic data is generated, mentioned in detail in a separate report named Synthetic_Anomalies_Generation_Report.pdf

4.2. Continuous Monitoring and Feedback

Given the dynamic nature of financial transactions, the anomaly detection system is designed to be continuously monitored:

- **User Feedback Loop:** When an anomaly is flagged, user responses (e.g., whether they decide to proceed or cancel the transaction) serve as an informal feedback mechanism.
- **Periodic Retraining:** The system architecture supports periodic retraining of the Isolation Forest model with new transaction data. This helps adapt to changes in spending patterns and ensures the model remains effective over time.
