**Credit Card Fraud Detection Model**

**Documentation**

**1. Introduction**

This document provides a detailed overview of the Credit Card Fraud Detection model, including its architecture, underlying logic, parameters, and implementation details.

**2. Model Overview**

The Credit Card Fraud Detection model is designed to identify potentially fraudulent transactions using machine learning techniques. The core of the model is a Random Forest Classifier, which processes transaction data to classify transactions into "low", "moderate" or "high" categories.

**3. Architecture**

**3.1 Data Generation**

To create a dataset for training and testing, synthetic data is generated using Python's Faker library. This includes:

* **Customer Data**: customer\_id, customer\_name, credit\_card\_number, customer\_latitude, customer longitude, and customer\_state.
* **Merchant Data**: merchant\_latitude, merchant longitude, and merchant\_state.
* **Transaction Data**: amount deducted, transaction\_id, transaction time, transaction\_date, and transaction\_hour.

**3.2 Prior Columns**

* **customer\_id**: Unique identifier for each customer.
* **customer\_name**: Name of the customer.
* **credit\_card\_number**: Credit card number used (should be handled securely).
* **amount deducted**: Amount deducted from the customer's account.
* **transaction\_id**: Unique identifier for each transaction.
* **customer\_latitude**: Latitude of the customer’s location.
* **customer longitude**: Longitude of the customer’s location.
* **merchant\_latitude**: Latitude of the merchant’s location.
* **merchant longitude**: Longitude of the merchant’s location.
* **customer\_state**: State where the customer is located.
* **merchant\_state**: State where the merchant is located.
* **transaction\_amount**: Total amount of the transaction.
* **transaction\_time**: Time of the transaction.
* **transaction\_date**: Date of the transaction.
* **transaction\_hour**: Hour of the transaction.

**3.2 Columns after model creation**

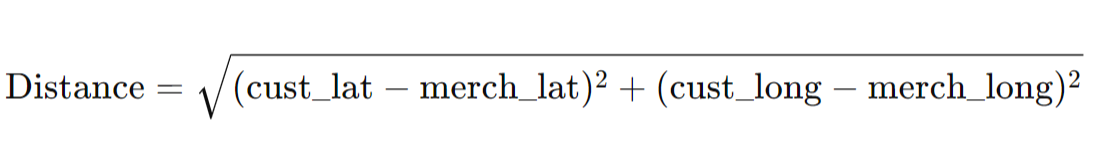
1. **unusual\_rating**: Rating indicating the unusual nature of a transaction.
2. **distance**: Distance between customer and merchant locations.
3. **distance\_rating**: Rating based on the distance of the transaction location from the customer's usual locations.
4. **state\_rating**: Rating based on the state where the transaction occurred, potentially reflecting regional trends or risks.
5. **limit\_rating**: Rating related to the transaction amount about predefined limits or thresholds.
6. **average\_rating**: Overall rating based on multiple factors or criteria.
7. **fraud\_risk**: Assessment of the likelihood that the transaction is fraudulent.Top of Form

Bottom of Form

**3.3 Feature Engineering**

Four key features are engineered from the raw data:

### 1. ****Distance****

**Description**: Measures how far apart the customer’s and merchant’s locations are. **Formula**: 

Assigns a rating based on the distance (in kilometers) between customer and merchant locations:

* **1**: Distance > 15,000 km
* **0.75**: Distance between 10,001 and 15,000 km
* **0.5**: Distance between 5,001 and 10,000 km
* **0.25**: Distance ≤ 5,000 km

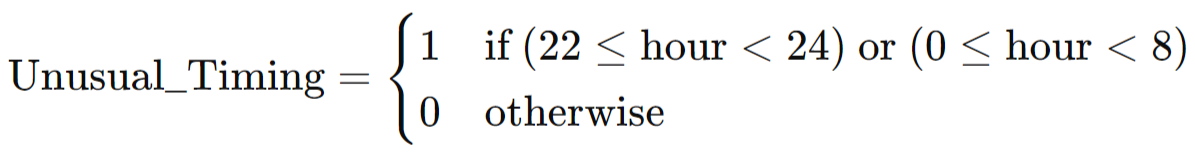
**Columns Used**:

* customer\_latitude
* customer\_longitude
* merchant\_latitude
* merchant\_longitude

### 2. ****Unusual Timings****

**Description**: Flags transactions that occur during late-night or early-morning hours, which might be less typical.

**Formula**:



### Explanation:

* **hour\text{hour}hour**: The hour of the transaction.
* **1**: Flags the transaction as unusual if it falls within the specified time ranges.
* **0**: Flags the transaction as normal if it does not fall within these time ranges.

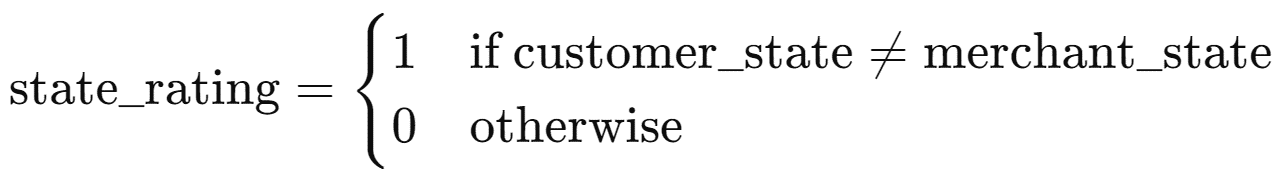
**Columns Used**:

* transaction\_hour

### 3. ****State****

**Description**: Indicates whether the transaction occurred in a different state from the customer’s usual state.

**Formula**:



### Explanation:

* **customer\_state**: State where the customer is located.
* **merchant\_state**: State where the merchant is located.
* **1**: Indicates that the transaction is in a different state.
* **0**: Indicates that the transaction is in the same state.

**Columns Used**:

* customer\_state
* merchant\_state

### 4. ****Frequency****

**Description**: Measures how often transactions occur within a specific region on a given date, helping identify unusual activity patterns.

**Calculation**:

* **Frequency**: The number of transactions for a given customer location and date.
* **Rating**: The frequency is divided by 5 and capped at 1. If the frequency exceeds 5, the rating is set to 1; otherwise, it is scaled proportionally.

The limit\_rating column represents how frequent transactions are at a particular location on a specific date, helping to identify unusual or high-activity patterns.

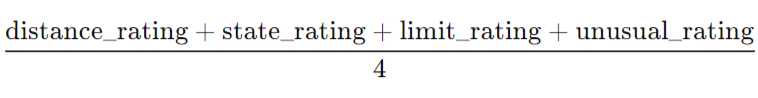
**Columns Used**:

* transaction\_date
* customer\_state or merchant\_state (depending on which region you are focusing on)

### 5. ****Average****

**Description**: Computes the average rating by aggregating scores from various factors: distance, state, frequency, and unusual timings.

**Calculation**:

average\_rating=

### Fraud Risk

**Description**: Classifies the fraud risk based on the average rating:

* **'low'**: Average rating between 0.0 and 0.4
* **'moderate'**: Average rating between 0.4 and 0.7
* **'high'**: Average rating of 0.7 and above

The fraud\_risk column categorizes transactions into risk levels based on their average rating, helping to assess the likelihood of fraudulent activity.

**3.4 Model Selection**

**Random Forest Classifier** is chosen due to its robustness in handling a variety of data types and its ability to manage overfitting through ensemble learning.

**4. Data Preparation**

**4.1 Backend Code**

pip install faker pandas numpy openpyxl geopy sklearn joblib

import pandas as pd

import numpy as np

from faker import Faker

import random

import string

from geopy.distance import great\_circle

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

import pickle

import joblib

### 4.2 Data Generation

* **Customer and Transaction IDs**: Randomly generated for uniqueness.
* **Transaction Amounts**: Uniformly distributed between 1 and 500.
* **Transaction Locations**: Random latitude and longitude values.

### 4.3 Data Augmentation

Duplicate transactions are introduced to simulate potential fraud cases.

### 4.4 Feature Engineering

* **Distance Calculation**: Using geopy to measure the distance between the customer and merchant.
* **Unusual Timing Rating**: Assigns a rating based on the transaction time.
* **Distance Rating**: Categorizes distance into ranges.
* **State Rating**: Flag for transactions in different states.
* **Frequency Rating**: Based on transaction frequency in specific groups of 100.

### 4.5 Model Training and Evaluation

* **Training**: The Random Forest model is trained using the processed dataset.
* **Evaluation**: Model performance is assessed using metrics like accuracy, confusion matrix, and classification report.

## 5. Model Implementation

### 5.1 Backend

The model is saved and loaded using joblib. A Flask application is set up to handle file uploads and processing.

### 5.2 Flask Application

**Overview:** This Flask application facilitates the upload and processing of Excel files containing transaction data to detect credit card fraud using a machine learning model.

**Features:**

* **File Upload:** Users can upload .xlsx files containing transaction data.
* **Data Processing:**
  + Calculates ratings based on unusual transaction times, distances between customer and merchant locations, state mismatches, and transaction frequency.k
  + Scales feature data and predict fraud risk using a pre-trained machine learning model.
* **Output:** Saves the processed data to a new Excel file with fraud risk predictions and provides a download link for the user.

**Routes:**

* **/**: Displays the file upload form (renders index.html).
* **/upload**: Handles file uploads, processes data, and returns the processed file for download.

**Configuration:**

* **Upload Folder:** Stores uploaded files.
* **Processed Folder:** Stores files with fraud analysis results.
* **Allowed Extensions:** Only .xlsx files are permitted.

**Dependencies:**

* Flask: Web framework for handling HTTP requests.
* pandas: Data manipulation and analysis.
* joblib: Model loading.
* werkzeug: Secure file handling.
* os: File and directory operations.

**Execution:**

* Runs a local web server in debug mode, creating necessary directories if they don't exist.

**5.3 Frontend**

**Overview:** This HTML code creates a web interface for a Credit Card Fraud Detection service, allowing users to upload Excel files for analysis. The system will use machine learning models to detect potential fraud in transaction data.

**Features:**

* **Title:** "Credit Card Fraud Detection" appears in the browser tab.
* **Container:** Centers and formats the content with a clean, user-friendly design.
* **Header:** Displays "Credit Card Fraud Detection" prominently.
* **Description:** Instructs users to upload their transaction data files for fraud analysis.
* **Form:** Includes a file upload field for .xls and .xlsx files and a submit button.
* **Footer:** Credits the service provider with a link to "OMFYS GROUP".

**Functionality:**

* Users upload Excel files for fraud detection analysis by the backend system.

**Style:**

* Utilizes modern design elements for a professional and intuitive user experience.

**6. Conclusion**

The Credit Card Fraud Detection model leverages machine learning and feature engineering to detect potential fraud in financial transactions. The model's accuracy and performance are validated, and a user-friendly web application is provided to facilitate real-time fraud detection.