**Feature Engineering Documentation for Synthetic Credit Fraud Dataset**

**1. Introduction**

This document outlines the feature engineering logic applied to a synthetic credit fraud dataset. The goal is to transform raw transactional data into meaningful features that can help detect fraudulent activities. The transformations include IP behavior analysis, temporal feature extraction, transaction pattern analysis, and categorical encoding.

**2. Feature Engineering Logic**



**2.1 IP Behavior Features**

**2.1.1 IP Country Code Extraction**

* **Logic:**
  + Fraudsters often use IPs from countries unrelated to the user’s profile (e.g., a UK customer suddenly transacting from Nigeria).
  + Helps flag geographic anomalies in transactions.
  + Maps each transaction’s IP address to its corresponding country code using the GeoLite2 database.
* **Function: add\_ip\_country\_code()**
  + Adds a new column IP\_address\_country\_code containing the 2-letter ISO country code.
  + Missing or invalid IPs return None.

**2.1.2 IP Timezone Conversion**

* **Logic:**
  + Detects timezone mismatches (e.g., transaction at 3 AM in the user's local time but 2 PM in the IP's country).
  + Converts transaction timestamps to the local timezone of the IP’s country.
  + Uses pytz to fetch timezone based on the country code.
* **Function: add\_ip\_country\_date()**
  + Outputs country\_code\_date (localized timestamp).
  + Drops intermediate timezone column.

**2.1.3 Distinct IP Count in Time Windows**

* **Logic:**
  + Legitimate customers often reuse the same IPs (static IPs), and a sudden IP change may be potentially a fraud indicator.
  + Tracks unique IPs per customer over rolling windows (1, 5, 12, 28, 60 days).
* **Function: add\_distinct\_ip\_window()**
  + For each rolling window, generates columns like distinct\_ip\_1d, distinct\_ip\_5d, etc.

**2.1.4 Shared IP Detection**

* **Logic:**
  + Fraudsters reuse IPs across multiple accounts (e.g., botnets).
  + Identifies IPs used by multiple customers (threshold: 1% of unique customers by default).
* **Function: add\_ip\_used\_by\_many()**
  + Adds binary columns (e.g., IP\_used\_by\_10\_customer\_id\_or\_more).

**2.1.5 Time Since Last IP Usage**

* **Logic:**
  + Calculates seconds since the same IP was last used by a customer.
  + New IPs being used may be a potential fraud flag
  + Sudden IP changes after long inactivity may indicate account takeover risk
* **Function: add\_time\_since\_last\_ip\_per\_customer()**
  + Outputs seconds\_since\_last\_ip (NaN filled with 0, for first IPs used).

**2.2 Temporal & Date-Based Features**

**2.2.1 Weekend Indicator**

* **Logic:**
  + Fraud transactions likely occur on holidays (Saturday/Sunday).
* **Function: add\_is\_weekend()**
  + Outputs binary column is\_weekend (1 if weekend).

**2.2.2 Weekday Indicators**

* **Logic:**
  + Creates separate binary flags for each weekday (e.g., is\_Monday).
* **Function: add\_is\_each\_weekday()**
  + Outputs 7 columns (Monday–Sunday).

**2.2.3 Bank Holiday Detection**

* **Logic:**
  + Fraud transactions likely occur on holidays.
  + Checks if transaction date is a UK bank holiday.
* **Function: add\_is\_UK\_bank\_holiday()**
  + Outputs is\_GB\_bank\_holiday (1 if holiday).

**2.2.4 Time Since Midnight**

* **Logic:**
  + Converts transaction time to seconds elapsed since midnight.
  + Unusual transaction times (e.g., 3 AM) may indicate higher risk.
* **Function: add\_total\_seconds\_since\_midnight()**
  + Outputs transaction\_date\_sec\_since\_midnight.

**2.3 Transaction & Merchant Features**

**2.3.1 Merchant Age**

* **Logic:**
  + Computes days since merchant establishment.
  + Newly created merchants are higher risk (fraudulent fronts, or lack of fraud knowledge to protect themselves).
* **Function: add\_days\_of\_established()**
  + Outputs established\_period\_in\_day.

**2.3.2 Transaction Moving Averages**

* **Logic:**
  + Calculates rolling averages transaction amount and transaction counts per customer.
  + Sudden spending spikes may indicate potential fraud.
* **Function: add\_ma\_trade\_info\_by\_id()**
  + For each rolling window, generates columns like customer\_id\_ma\_amount\_5d.

**2.3.3 Merchant Mismatch Flags**

* **Logic:**
  + Detects mismatches between merchant\_id and store\_card\_merchant\_id.
  + Sudden spending spikes indicate potential fraud.
* **Function: add\_merchant\_card\_flags()**
  + is\_merchant\_card (1 for store credit cards).
  + is\_merchant\_mismatch (1 if IDs of store card mismatch with transaction merchant IDs).
  + is\_merchant\_card\_and\_mismatch (1 if both of above are 1)

**2.4 Categorical Encoding**

**2.4.1 One-Hot Encoding**

* **Logic:**
  + Converts categorical data (e.g., merchant\_category) into model-digestible numeric features
* **Function: onehot\_encode\_column()**

**2.4.2 Label Encoding**

* **Logic:**
  + Converts strings (e.g., card\_present\_or\_not) to numerical labels for model to digest.
* **Function: label\_encode\_column()**

**3. Output Features**

The final engineered features include:

* **IP Behavior:**  
  IP\_address\_country\_code, distinct\_ip\_{n}d, IP\_used\_by\_{threshold}\_customer\_id\_or\_more, seconds\_since\_last\_ip
* **Temporal:**  
  is\_weekend, is\_{weekday}, is\_UK\_bank\_holiday, transaction\_date\_sec\_since\_midnight
* **Transaction Patterns:**  
  customer\_id\_ma\_amount\_{n}d, customer\_id\_ma\_count\_{n}d, is\_merchant\_card
* **Categorical Encodings:**  
  merchant\_category\_{dummies}, card\_present\_or\_not (label encoded)