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.

Due to interpretability concerns, traditional logical machine learning models are the preferred choice. Consequently, higher-level data and temporal structures need to be discretized into scalar fields to ensure compatibility with these models.

---

2. Feature Engineering Logic

Feature Category	Subcategory	Original Feature Name	Original Type	Transformation Function	Transformation Description	New Features Count	New Feature Type
Customer	*IP behavior*	IP_address	string	`add_ip_country_code()`	Mapped to ISO country code	1	categorical
				`add_ip_country_date()`	Timezone conversion	1	datetime
				`add_distinct_ip_window()`	Distinct IP counts (1d,5d,12d,28d,60d)	5	numeric
				`add_ip_used_by_many()`	Shared IP detection	1+	binary
				`add_time_since_last_ip_per_customer()`	Seconds since last IP usage	1	numeric
	Transaction	transaction_date	datetime	`decompose_date_components()`	Extracted day/month	2	numeric
				`add_total_seconds_since_midnight()`	Seconds since midnight	1	numeric
		amount	numeric	`add_ma_trade_info_by_id()`	Rolling averages (1d,5d,12d,28d,60d)	5	numeric
				`add_ma_trade_info_by_id()`	Rolling transaction counts	5	numeric
Merchant	*Attributes*	merchant_category	categorical	`onehot_encode_column()`	One-hot encoded categories	Varies	binary
		merchant_established_date	datetime	`add_days_of_established()`	Days since establishment	1	numeric
	Card Relations	type_of_credit_card_used	categorical	(Manual)	Created: is_merchant_card, is_merchant_mismatch, is_merchant_card_and_mismatch	3	binary
		store_card_merchant_id	categorical	`combine_two_cols_with_null_check()`	Combined with credit card type	1	categorical

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).
    - Turn IP address into facts of IP address distribution for model compatibility
- **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).
    - Boolean fields about the current timestamp to capture seasonality.
- **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).
    - Boolean fields about the current timestamp to capture seasonality.
- **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)
