hypertrack-v1.py

hypertrack-v1.py is the first version of hypertrack used to identify bad couriers and deactivate their accounts. It can read infinite number of rules from *rules-v1.json* file, which is the format used to create different rules. Each rule has different variables, including Pinot query, Pinot database port number, region IDs, time threshold and instance threshold. In this way, the original script has been expanded to handle a whole variety of rules, which run one after the other. The details of each function are introduced below:

1. The `*excute_rule*` function takes variables from rule and run put these variables in other functions
2. The `*failed_orders_in_time*` function query from Pinot to get the failed orders information through desired query in desired database within desired region and time frame
3. The `*unique_couriers_failing_orders*` function is executed to get unique couriers from failed orders, and it remains unchanged
4. Variable `*instance_threshold*` is added to `*identify_really_bad_couriers*` function in order to identify bad couriers using rule of with different threshold
5. The `*waitlist_really_bad_couriers*` function deactivates bad couriers and it remains unchanged

In this way, a hard-coded script supporting only one rule is transformed to a powerful platform for data analysts and scientist to change and execute different real-time fraud rules.

The v1 meets basic requirements of the assigned task. However, the rules may not work well in detecting fraud in long-term base, for example, part-time couriers who deliver only 1 or 2 orders every day but abuse occasionally by failing most of the orders. In order to detect different kind of fraud cases, I created v2 with new rule.

hypertrack-v2.py

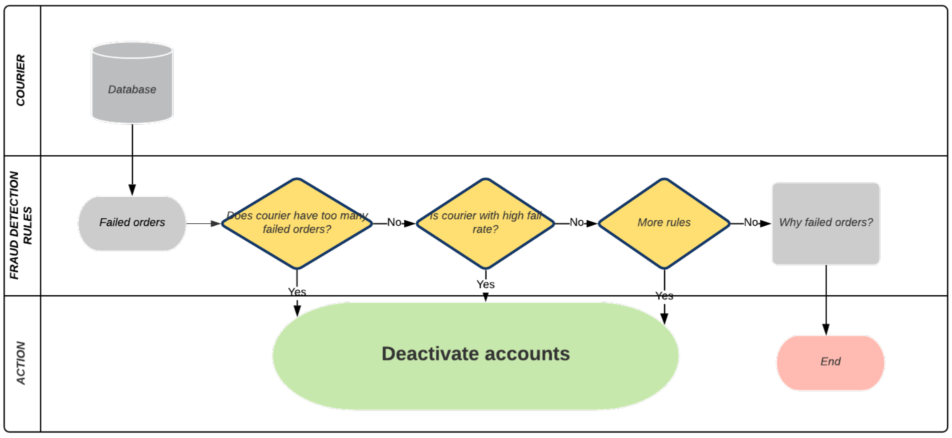
Instead of calculating number of failed orders in certain amount of time, v2 calculates fail rate for each courier, which is defined as:

$$\text{fail rate} = \frac{\text{\# of failed orders}}{\text{\# of all orders}}$$

Similar to v1, v2 reads infinite number of rules from *rules-v2.json*, which takes different variables including Pinot query, Pinot database port number, region IDs, time threshold and fail rate threshold. And here are some differences:

1. Instead of time threshold in minutes, `*failed_orders_in_time*` in v2 takes time threshold in days, because we are interested in fail rate in longer period
2. A new function `*get_couriers_order_count*` is defined, which returns a dictionary containing count of total order for these bad couriers in the same time frame through Query Pinot from a table containing data for all orders
3. The `*identify_really_bad_couriers*` function has been changed to calculate fail rate, and then this is compared with certain threshold to detect really bad couriers

This flow chart below illustrates how rules are evaluated and action is triggered:



For couriers who have failed orders but are not labeled as bad users, we can either create more rules or investigate into the reasons behind failed orders and make improvement in the future. In this way, this powerful platform check system can identify frauds using variety of rules and have massive business impact on Uber Eats.