Report

1. Mathematical model

Consider a state $s \in S$ from the set of all possible clients' states, describing the client at any moment in time. For example, S might be the set of real valued n–dimensional vectors. We will introduce three functions f_{dialog}, f_{geo}, $f_{transaction}: S \to S$ which are going to update the client's state after any respective action has been performed by the client. If no information about the client was given prior, their state is set to s_0 by default (e. g. null vector if $S=R^n$). In each state the client has some probability of purchasing one of four target products. This probabilities can be described by a different function $P_i: S \to [0; 1]$, where the value of $P_i(s)$ represents the probability of buying a target product *target_i* a client being in the state *s*. If probability is greater than 0.5, we assign a positive class to the target and negative otherwise.

In order to predict whether or not a client would purchase some of four rather popular products in the next month, we will have to find a set of functions f_{dialog}, f_{geo}, $f_{transaction}$ and P_i for $i \in \{1, 2, 3, 4\}$ such that the results would match the real data.

2. Workflow

We begin with inspecting each table and preparing the data: number of features and samples in the dataset, which features are numerical, and which are categorical, null values, why they occur and how should we fix them, types of data. We then try to learn about the distributions of features by drawing diagrams and performing statistical tests. During this we look for outliers, pick up on the patterns and try to formulate hypotheses.

To prepare the data before training the machine learning model it is necessary to encode categorical features (e.g. using one hot or ordinal encoding). It is also highly recommended to normalize numerical features first. Given the nature of the data we need to account for the time of events: the model should only learn from the data from the past, which makes the task significantly more difficult. It is also important to notice a huge imbalance in the data: there is only a tiny proportion of samples of positive class in any target feature (less than 0.42%). To fight this I have chosen to manually limit the number of negative classes in the training data in order for it to be on par with the other category.

Earlier I have already described the mathematical model that might be capable of solving the task at hand, although many details of actually implementing it are still unclear to me. After several not so fruitful attempts of using *logistic regression* and *random forest* (getting a value of *recall* around 0.51-0.56), it have come to the conclusion that a more sophisticated model architecture is required, perhaps a *multi-layer perceptron* or even a *transformer*, (which I have heard is ideal for my idea of updating the state vector after every event, but I am clearly lacking the knowledge in this area).