**INTRODUCTION**

Credit card fraud detection is a relevant problem that draws the attention of machine-learning and computational intelligence communities, where a large number of automatic solutions have been proposed. In fact, this problem appears to be particularly challenging from a learning perspective, since it is characterized at the same time by class imbalance, namely, genuine transactions far outnumber frauds, and concept drift, namely, transactions might change their statistical properties over time. These, however, are not the only challenges characterizing learning problems in a real-world fraud-detection system (FDS).

In a real-world FDS, the massive stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize. Classifiers are typically employed to analyze all the authorized transactions and alert the most suspicious ones. Alerts are then inspected by professional investigators that contact the cardholders to determine the true nature (either genuine or fraudulent) of each alerted transaction. By doing this, investigators provide a feedback to the system in the form of labeled transactions, which can be used to train or update the classifier, in order to preserve (or eventually improve) the fraud-detection performance over time. The vast majority of transactions cannot be verified by investigators for obvious time and cost constraints. These transactions remain unlabeled until customers discover and report frauds, or until a sufficient amount of time has elapsed such that non-disputed transactions are considered genuine.

Thus, in practice, most of supervised samples are provided with a substantial delay, a problem known as verification latency. The only recent supervised information made available to update the classifier is provided through the alert– feedback interaction. Most papers in the literature ignore the verification latency as well as the alert–feedback interaction, and unrealistically assume that the label of each transaction is regularly made available to the FDS, e.g., on a daily basis. However, these aspects have to be considered when designing a real-world FDS, since verification latency is harmful when concept drift occurs, and the alert–feedback interaction is responsible of a sort of sample selection bias (SSB) that injects further differences between the distribution of training and test data.