Self-Adaptive Financial Fraud Detection System

Submitted by: Team 9 – Manan Raheja & Aman Kumar, Department of Electrical and Computer Engineering, University of Waterloo

Motivation

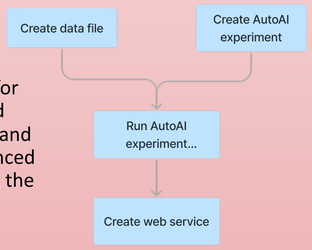
- Financial fraud detection is a challenging problem due to data drift, with serious consequences.
- Traditional (static) ML models easily become outdated.
- Automating the process of model retraining keeps it relevant.

Objective of the Project

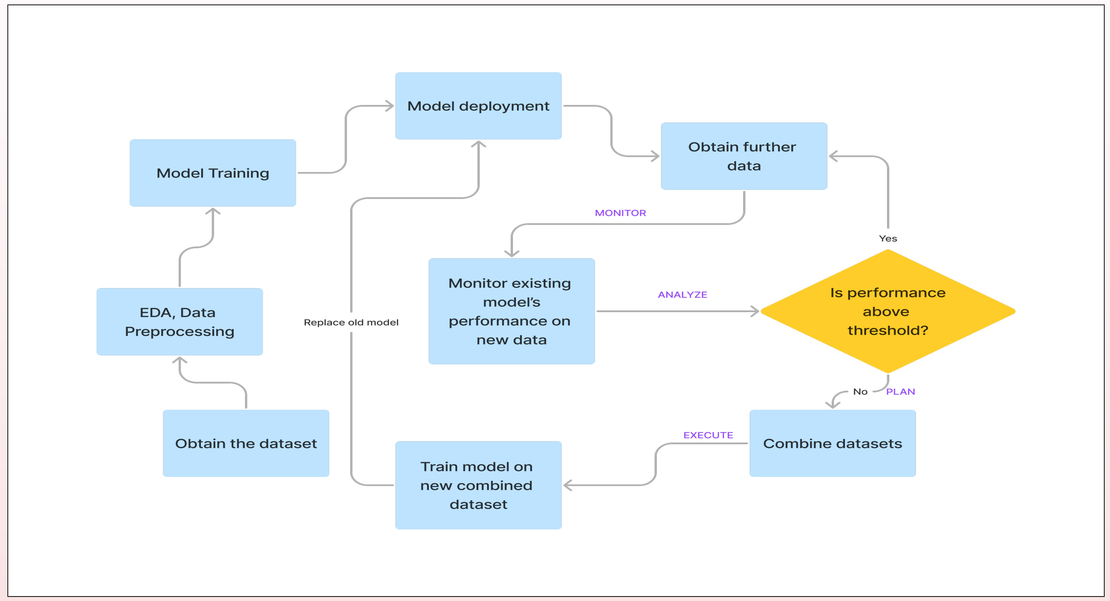
- To build a fraud detection system that can adapt to the data drift in the financial transactions
- Obtain and preprocess financial transaction dataset.
- To train and deploy an ML model.
- Make it self-optimizing by MAPE-K loop on MLOps.

Proposed Solution

- We present an implementation of a self-adaptive financial fraud detection system, that integrates a MAPE-K loop into an MLOps process, enabling continuous monitoring of data drift and autonomous adaptation to emerging patterns.
- By dynamically updating and retraining the machine learning model, the proposed system effectively maintains high-performance levels, even in the face of significant data drift.
- **Two experiments:**
  - **Online Experiment:** Here we used IBM Cloud Pak for Data to create project, IBM Watson Machine Learning to make pipeline to train and deploy the model and IBM Cloud Object Storage for storing assets.
  - **Offline Experiment:** Here we replicated the same process as for online variant, however we used imbalanced dataset for training and sharded and streamed the balanced dataset for testing and updating the model on local system.



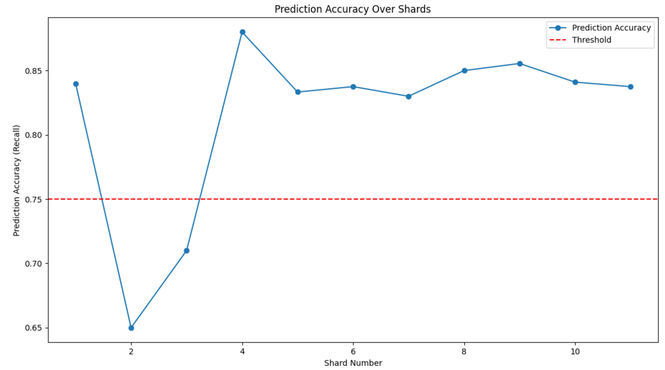
Our Approach: MAPE-K Loop + MLOps

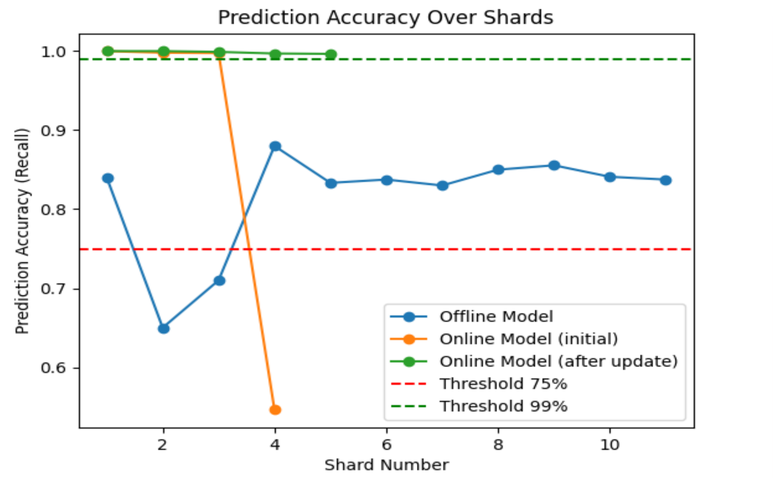


Components of the Architecture

- **Data Streamer**: This component simulates the continuous data stream of verified transactions to create scenarios as close as possible to real-world conditions.
- **Model Training & Deployment Pipeline:** It takes the transaction data as input.
  - **Step 1:** it performs the data engineering steps including data transformation, data cleaning and data normalization, to prepare the data for model training.
  - **Step 2:** Then, it trains a new custom voting classifier model on the latest data and updates the deployment with the new model.
- **Driver Program**: This is the main component that **orchestrates the complete self-adaptive processes**: MAPE-K loop and the MLOps processes.
  - **Step 1:** This component would constantly monitor the ML model's performance on the stream of data.
  - **Step 2:** If the performance is above a set threshold, then the system will loop back to Step 1 as the data has not experienced significant drift, the model is performing well as per the requirements and we do not need to waste resources in re-training the model. Else, go to Step 3.
  - **Step 3:** If the performance on the new data is below the threshold, the system would include the new data in the current dataset and it would trigger the model training pipeline to re-train the model on the updated dataset.
  - **Step 4:** The pipeline will re-train the model on the combined dataset and then deploy the re-trained model.

Results





Offline Experiment (contd):

- Built a custom voting classifier combining XGBoost, Random Forest and Gaussian Naïve Bayes.
- The observations underscore the efficacy of this self-adaptive framework, showcasing its ability to sustain optimal performance and adaptability amidst the ever-changing landscape of financial transaction data.

Classifier	Recall
AdaBoost	74.48%
Bagging Classifier	80.61%
BernoulliNB	63.26%
Calibrated Classifier CV	63.26%
DecisionTreeClassifier	75.51%
ExtraTreeClassifier	76.53%
ExtraTreesClassifier	82.64%
GaussianNB	84.69%
KNeighborsClassifier	80.61%
XGBoost	82%
LinearSVC	78%
SVC	67%
Random Forest	81%
Logistic Regression	65%
Gradient Boosting (max_depth=15)	78%
Custom Voting Classifier	83%

TABLE I: Performance of different classifiers on the Credit Fraud Dataset