A Real-Time Monitoring Framework for Deployed Machine Learning Models with Drift Detection and Performance Analytics

Ganga Vijay Kumar, Ms.B.Pramodhini
M.Sc Artificial Intelligence & Data Science Department of Computer Science & AI Central University of Andhra Pradesh
Assistant Professor Department of Computer Science & AI Central University of Andhra Pradesh
doi.org/10.64643/IJIRTV12I12-201193-459

***Abstract*—Machine learning models have been broadly de-ployed in real world application, for example, making predictive decisions.However, performance of the deployed model degrades rapidly with time since data distribution will change. The phenomenon is widely known as data drift or concept drift and cause problems for a deployed model. In this paper, we introduce a monitoring framework designed for the real time evaluation and sustainment of a deployed machine learning model in a changing world.A customer churn dataset which contains 7032 samples and 31 features was utilized to train a set of machine learning models, which include Logistic Regression, Random Forest and XGBoost. And they are split into train set (80%) and test set (20%). Random Forest is chosen among them as the final model, which has the accuracy of 79.45% and is deployed in a FastAPI based service to mimic real time prediction. Our system continuingly record the prediction logs, evaluates and monitors the performance trend of the model and employs PSI (Population Stability Index) to capture distribution drift for features. An interactive dash board with Streamlit is also provided to visualize the behavior, performance and drift features of model. The result from experimentation confirms that our system can effectively detect the distribution shift of features (e.g. Tenure and billing attributes), which represents the possibility of model performance degradation at the early stage. This work focuses on post-deployment monitoring system and can be a practical approach for a real world application to guarantee a sustainable performance for machine learning models.**

***Index Terms*—Machine Learning, Model Monitoring, Data Drift, Population Stability Index, Random Forest, Real-Time Systems**

I. INTRODUCTION

Machine learning models (ML) are being used more and more in real-life systems to enable automation of decisions in telecommunication, finance, medicine, e-commerce and other areas [1], [2]. Most ML models are trained using past data and validated using known metrics before deploying to the production environment. While this validation gives an idea of how well a model is performing, there is no assurance that it will continue performing at the same level once it encounters a changing real-world environment.

In the real-world environment, distributions of data change all the time due to many reasons like changing user behavior, seasons etc. A model trained with static data loses its predictive ability over time. These kinds of performance degradations are attributed to data drift, where the statistical characteristics of features change with time and concept drift where relationship between features and the target variable change [3], [4]. If not monitored, these changes lead to errors by the model and subsequent incorrect decisions.

Post-deployment monitoring has become an overlooked component of many machine learning applications despite of widespread usage of machine learning systems [2]. Most ML applications follow a workflow which is primarily focused on model development and testing only with no attention on its continuous performance evaluation after deployment. This implies that models may be deployed without any monitoring systems which provide an overview of performance degradation and input distribution changes. The disparity is evident in how ML systems are currently being developed.

There is research available regarding monitoring methods that are efficient for data distribution changes. Nevertheless, there is limited work where the system has fully integrated components

for both algorithmic drift detection and full system-based deployment and visualization. It is necessary that the system provides decision support by performing both the detection of the drift and presenting interpretable information. In this paper, we propose a real-time monitoring framework for deployed machine learning models. Our system provides fully integrated system for training, API based deployment, continuous monitoring and statistical drift detection with in-teractive visualizations. We use a customer churn data set for training several ML models and deploy the best model on a FastAPI based service to demonstrate real-time inference.

In this paper we provide an integrated production ready system that covers model serving, continuous performance evaluation, feature-level drift detection using statistical techniques, and interactive analytics enabling quick detection of model performance degradation.

A. Significance of Contributions

The key contributions of the proposed work are summarized as follows:

- An End-to-End monitoring framework: A holistic framework that combines training, deployment, real-time monitoring, and visualization together to address the ”gap” between models in development and models in real world applications.
- Real-time model monitoring: A continuous analysis of the model’s predictions and their trend can be carried out based on a FastAPI-based deployment to understand model performance under simulated real-time.
- Drifting detection mechanism: The implementation of a statistical method (Population Stability Index (PSI)) to analyze distribution changes in features and potential degradation in model performance.
- Interactive visualization dashboard: An easy-to-use dashboard that leverages Streamlit and shows the distribution of predictions, monitoring statistics and drifting alert to users.
- Deployment-oriented design: Contrary to the common theoretical approach, this system is built around practical deployment, with real-time inference, monitoring and alerting mechanisms integrated with the machine learning models.

II. LITERATURE REVIEW

The problem of fraud detection and model reliability in dynamic environments has been extensively studied across both academia and industry. Existing approaches can be broadly categorized into traditional rule-based systems, machine learning-based detection models, and recent advancements in model monitoring and drift detection techniques.

A. Rule-Based and Statistical Approaches

Early fraud detection systems relied heavily on rule-based mechanisms and statistical thresholds derived from historical transaction patterns. These systems operate by defining expert-driven rules to flag anomalous activities. While such approaches are computationally efficient and interpretable, they lack adaptability to evolving fraud patterns and often suffer from high false positive rates. Moreover, maintaining rule sets becomes increasingly complex as transaction volumes grow [5].

B. Machine Learning-Based Detection

With the advancement of data-driven methods, machine learning models such as Logistic Regression, Decision Trees, Random Forests, and Gradient Boosting have been widely adopted for fraud detection tasks [6], [7], [8]. These models learn complex relationships between transaction features and fraud labels, improving detection accuracy compared to rule-based systems.

Ensemble methods, particularly Random Forest and XG-Boost, have demonstrated strong performance due to their ability to capture non-linear patterns and handle high-dimensional data. However, these models are typically trained on static datasets and assume that future data distributions remain consistent with training data. This assumption often fails in real-world scenarios, leading to performance degradation over time.

C. Concept Drift and Model Monitoring

Recent research has focused on the impact of concept drift, where the statistical properties of input data change over time. As highlighted in [9], drift can be formally expressed as a change in the joint probability distribution:

$$P_t(X, y) \neq P_{t+\Delta t}(X, y)$$

Drift detection techniques are generally classified into super-vised and unsupervised methods.

Supervised approaches rely on labeled data to monitor performance degradation, whereas unsupervised methods detect distributional changes without requiring ground truth labels. Statistical techniques such as Kolmogorov-Smirnov tests, Population Stability Index (PSI), and distribution divergence measures are commonly used for detecting drift in streaming data [3], [4], [5].

Although these methods are effective in identifying distribu-tion changes, many existing solutions focus solely on detection and do not provide an integrated framework for deployment, monitoring, and visualization.

D. Deployment and Monitoring Systems

In recent years, there has been growing interest in inte-grating machine learning models with deployment frameworks such as REST-based APIs and real-time dashboards. These systems aim to bridge the gap between model development and production environments by enabling real-time predictions and monitoring.

However, most existing implementations treat model pre-diction, monitoring, and visualization as separate components. This lack of integration limits the ability to perform continuous evaluation and rapid response to performance degradation.

E. Research Gap

Despite significant advancements in fraud detection and drift analysis, several challenges remain unaddressed:

- Most existing models are designed for offline evaluation and do not support continuous monitoring in real-time environments.
- Drift detection techniques are often implemented inde-pendently and are not integrated with deployed machine learning systems.
- Limited emphasis is placed on combining prediction, monitoring, drift detection, and visualization within a unified framework.
- Existing systems lack practical deployment-oriented de-signs that can operate efficiently in real-time scenarios.

To address these limitations, this work proposes an inte-grated machine learning monitoring framework that combines model prediction, performance tracking, drift detection, and interactive visualization within a single system.

III. METHODOLOGY

The proposed system is designed as an end-to-end frame-work that integrates data preprocessing, model development, deployment, and continuous monitoring. The primary objec-tive is to ensure not only predictive accuracy but also sustained reliability of the model in dynamic environments where data distributions may evolve over time.

A. System Overview

The overall architecture consists of multiple interconnected components, including data preprocessing, model training, API-based deployment, performance monitoring, and drift detection. The system operates in a sequential pipeline where input data is processed, predictions are generated, and moni-toring mechanisms continuously evaluate model behavior.

Let $X \in R^n$ represent the input feature vector and $y \in \{0, 1\}$ denote the target variable. The trained model $f(X)$ produces a prediction $\hat{y}$ and an associated probability score $P(y|X)$:

$$\hat{y} = f(X), \quad P(y|X) = \text{model confidence}$$

This formulation allows the system to not only classify outcomes but also quantify prediction uncertainty.

B. Data Preprocessing

The dataset consists of 7032 instances with 31 features representing customer demographics, service attributes, and billing information. Preprocessing is performed to ensure data quality and consistency.

Missing values are handled to prevent incomplete data from affecting model performance. Categorical variables are transformed into numerical representations using encoding techniques, while numerical features are scaled to maintain uniformity across feature ranges. This step ensures that all features contribute proportionally during model training.

The processed dataset is then divided into training and testing sets using an 80:20 split, enabling reliable evaluation of model performance.

C. Model Training and Selection

Multiple machine learning models are trained, including Logistic Regression, Random Forest, and XGBoost [6], [7], [8]. Each model is evaluated using standard classification metrics such as

accuracy, precision, recall, and F1-score.
Accuracy is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP , TN , FP , and FN represent true positives, true negatives, false positives, and false negatives, respectively.
Among the evaluated models, Random Forest demonstrated superior performance and stability. Due to its ensemble nature and ability to capture non-linear relationships, it was selected as the final model for deployment.

D. Model Deployment

The selected model is deployed using a FastAPI-based service, enabling real-time prediction through RESTful API endpoints. Incoming data instances are processed individually, and the model generates predictions along with probability scores.
Each prediction event is logged with corresponding input features and timestamps. This logging mechanism enables continuous tracking of model behavior and supports further analysis in the monitoring phase.

E. Performance Monitoring

To evaluate model performance after deployment, prediction data is continuously collected and analyzed. Instead of relying solely on static evaluation metrics, the system tracks prediction trends over time.
Let $\hat{y}_t$ denote predictions at time t. The system evaluates performance as a function of time:

$$Performance(t) = g(\hat{y}_t, y_t)$$

where $g(\cdot)$ represents evaluation metrics such as accuracy or recall computed over a time window.
This approach enables detection of gradual performance degradation that may not be visible in static evaluations.

F. Drift Detection

To identify changes in data distribution, the Population Stability Index (PSI) is used as a statistical measure. PSI compares the distribution of input features in the training data (baseline) with incoming data.
The PSI is defined as:

$$PSI = \sum (Actual\% - Expected\%) \cdot \ln \frac{Actual\%}{Expected\%}$$

where Expected% represents the proportion of observations in each bin from the baseline dataset, and Actual% represents the proportion from incoming data.
Higher PSI values indicate significant deviation between distributions, signaling potential data drift. In the proposed system, features such as tenure, monthly charges, and total charges are monitored, and high PSI values trigger alert conditions.

G. Visualization and Alert System

To enhance interpretability, a Streamlit-based dashboard is developed to visualize prediction behavior, performance met-rics, and drift indicators. The dashboard presents key insights such as prediction distribution, churn ratio, and probability trends.
Additionally, an alert mechanism is integrated to notify users when significant drift is detected. This enables timely intervention and supports maintaining model reliability in production environments.

IV. RESULTS AND DISCUSSION

The proposed framework was evaluated using the customer churn dataset consisting of 7032 instances and 31 features. Multiple machine learning models were trained and compared using an 80:20 train-test split to identify the most suitable model for deployment.

A. Model Performance Comparison

The performance of Logistic Regression, Random Forest, and XGBoost models was evaluated using standard classification metrics. The results are presented in Table I.

TABLE I: MODEL PERFORMANCE COMPARISON

Model	Accuracy
Logistic Regression	78.74%
Random Forest	79.45%
XGBoost	78.25%

Among the evaluated models, Random Forest achieved the highest accuracy and demonstrated stable performance. This can be attributed to its ensemble learning capability, which allows it to capture complex relationships in the data while reducing variance. Logistic Regression showed slightly lower performance due to its linear nature, while XGBoost performed comparably but did not surpass Random Forest under the given configuration.

B. Real-Time Monitoring and Dashboard Analysis

The deployed model was integrated with a FastAPI-based service to simulate real-time prediction. The monitoring dash-board provides a comprehensive overview of model behavior, including prediction distribution and key performance indica-tors.

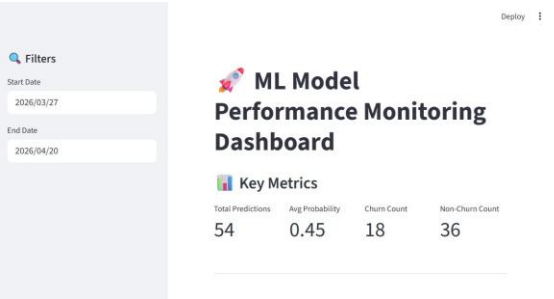

Fig. 1. Real-Time Monitoring Dashboard Showing Prediction Distribution and Key Metrics

As shown in Fig. 1, the dashboard visualizes total pre-dictions, churn ratio, and average probability. This enables continuous tracking of model outputs and supports better interpretability of predictions.

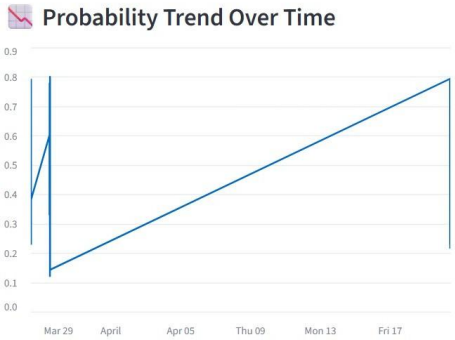

Fig. 2. Trend of Prediction Probability Over Time

C. Prediction Probability Analysis

To understand model confidence over time, prediction prob-abilities were analyzed across sequential inputs.

The probability trend illustrated in Fig. 2 shows how the model's confidence evolves with incoming data. Variations in probability values indicate changing input patterns and highlight the importance of continuous monitoring in real-time systems.

D. Drift Detection Analysis

A key component of the proposed framework is the detection of data drift using the Population Stability Index (PSI). The system compares feature distributions between baseline training data and incoming data to identify deviations.

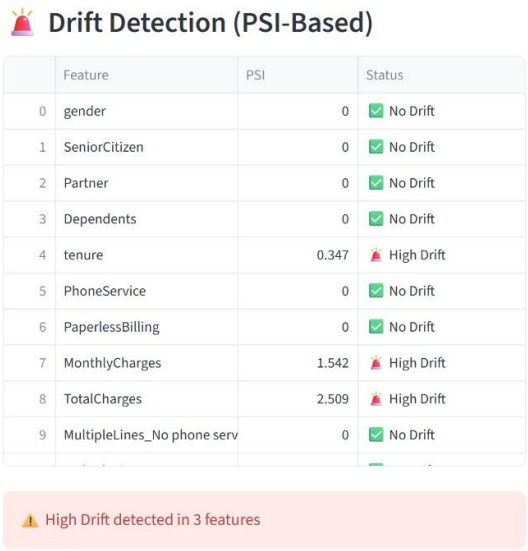

Fig. 3. Feature-Level Drift Detection Using Population Stability Index (PSI)

As shown in Fig. 3, features such as tenure, monthly charges, and total charges exhibit significant PSI values, indicating noticeable drift. These deviations trigger alert conditions, demonstrating the system's ability to detect distributional changes effectively [9], [3].

E. Discussion

The experimental results demonstrate that the proposed framework not only achieves reliable predictive performance but also provides effective monitoring and drift detection capabilities. The integration of real-time prediction, statistical analysis, and visualization ensures that changes in data distribution can be identified early.

Although the overall model accuracy is moderate, the sys-tem's strength lies in its ability to maintain awareness of model behavior after deployment. This is particularly important in real-world applications where data is continuously evolving and static evaluation is insufficient.

V. CONCLUSION

This paper presented a real-time monitoring framework for deployed machine learning models, focusing on maintaining reliability in dynamic environments. The proposed system integrates data preprocessing, model training, API-based deployment, continuous performance monitoring, drift detection, and visualization into a unified pipeline.

Experimental results on the customer churn dataset demon-strated that the Random Forest model achieved the best performance with an accuracy of 79.45%. More importantly, the system successfully monitored prediction behavior over time and detected feature-level data drift using the Population Stability Index (PSI). The integration of an interactive dashboard further improved interpretability by providing clear insights into prediction trends and drift conditions.

The results highlight that, beyond predictive accuracy, continuous monitoring is essential for ensuring the long-term effectiveness of machine learning systems in real-world applications. The proposed framework addresses this need by enabling early detection of performance degradation and supporting informed decision-making.

Future work may focus on incorporating automated model retraining mechanisms, advanced drift detection techniques, and scalability improvements for handling large-scale stream-ing data. Additionally, extending the framework to support multiple models and adaptive learning strategies can further enhance its applicability in production environments.

A. Future Work

Future work will focus on enhancing the adaptability and scalability of the proposed framework. One possible extension is the integration of automated model retraining mechanisms that can be triggered when significant drift is detected, enabling the system to maintain performance without manual intervention.

In addition, more advanced drift detection techniques, such as distribution-based statistical tests and machine learning-driven drift detectors, can be incorporated to improve detection accuracy and robustness. Extending the framework to support streaming data architectures will further enable real-time processing of high-velocity data in large-scale production environments.

Another direction involves incorporating explainability techniques to better interpret model predictions and drift behavior, which is particularly important for decision-critical applications. Finally, the framework can be extended to support multi-model monitoring and adaptive model selection strategies, allowing dynamic switching between models based on performance and data conditions.

REFERENCES

[1] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagap-pan, B. Nushi, and T. Zimmermann, “Software engineering for machine learning: A case study,” in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2019.

[2] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, “The ml test score: A rubric for ml production readiness,” in *Proceedings of the IEEE International Conference on Big Data*, 2017.

[3] J. Gama, I. Zˇ liobaite˙, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, 2014.

[4] A. Bifet and R. Gavalda`, “Learning from time-changing data with adap-tive windowing,” in *Proceedings of the SIAM International Conference on Data Mining*, 2007.

[5] N. Siddiqi, *Credit Risk Scorecards: Developing and Implementing Intel-ligent Credit Scoring*. Wiley, 2006.

[6] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. Wiley, 2013.

[7] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the ACM SIGKDD International Conference on Knowl-edge Discovery and Data Mining*, 2016, pp. 785–794.

[9] D. M. P. *et al.*, “Driftlens: Continuous drift detection and explainability,” *IEEE Transactions on Knowledge and Data Engineering*, 2023.