PROJECT: Fraud detection in a government agency (spotlight: MLOps)

Tourad Baba

Date: 2024-02-14

GitHub Link:
https://github.com/TouradBaba/Fraud_det_model_MLOps

APP URL : https://frauddetectionmodel.azurewebsites.net

Table of content

Introduction

Overview of the Project:

The project focuses on the integration of predictive model into a service, showcasing the practical application of machine learning in real-world scenarios.

Significance of Integration:

Integrating a predictive model into an application brings the power of data-driven decision-making.

This enhances the application's ability to make informed predictions, automate processes, and adapt to dynamic conditions.

Challenges Addressed:

Throughout the integration process, I have encountered various challenges that required innovative solutions.

This presentation delves into the complexities faced during the integration, highlighting the strategies implemented to overcome these challenges.

Project Overview

Summary of the Predictive Model:

The predictive model is a sophisticated implementation of machine learning algorithm tailored for fraud detection.

Leveraging techniques such as logistic regression, the model is trained on a comprehensive dataset to identify patterns indicative of fraudulent financial transactions.

Rigorous evaluation ensure high accuracy, making it a robust tool for real-time fraud prevention.

Description of the Application/Service:

The predictive model seamlessly integrates into an application built with Flask Library, providing a comprehensive solution for fraud detection.

Users can interact with the system through an intuitive interface, gaining valuable insights into transaction legitimacy and receiving immediate alerts for potential fraud.

The user-friendly interface supports both manual entry of transaction details and batch processing using file uploads, enhancing flexibility and usability.

Data Acquisition and Preprocessing

Data Requirements for Model Training

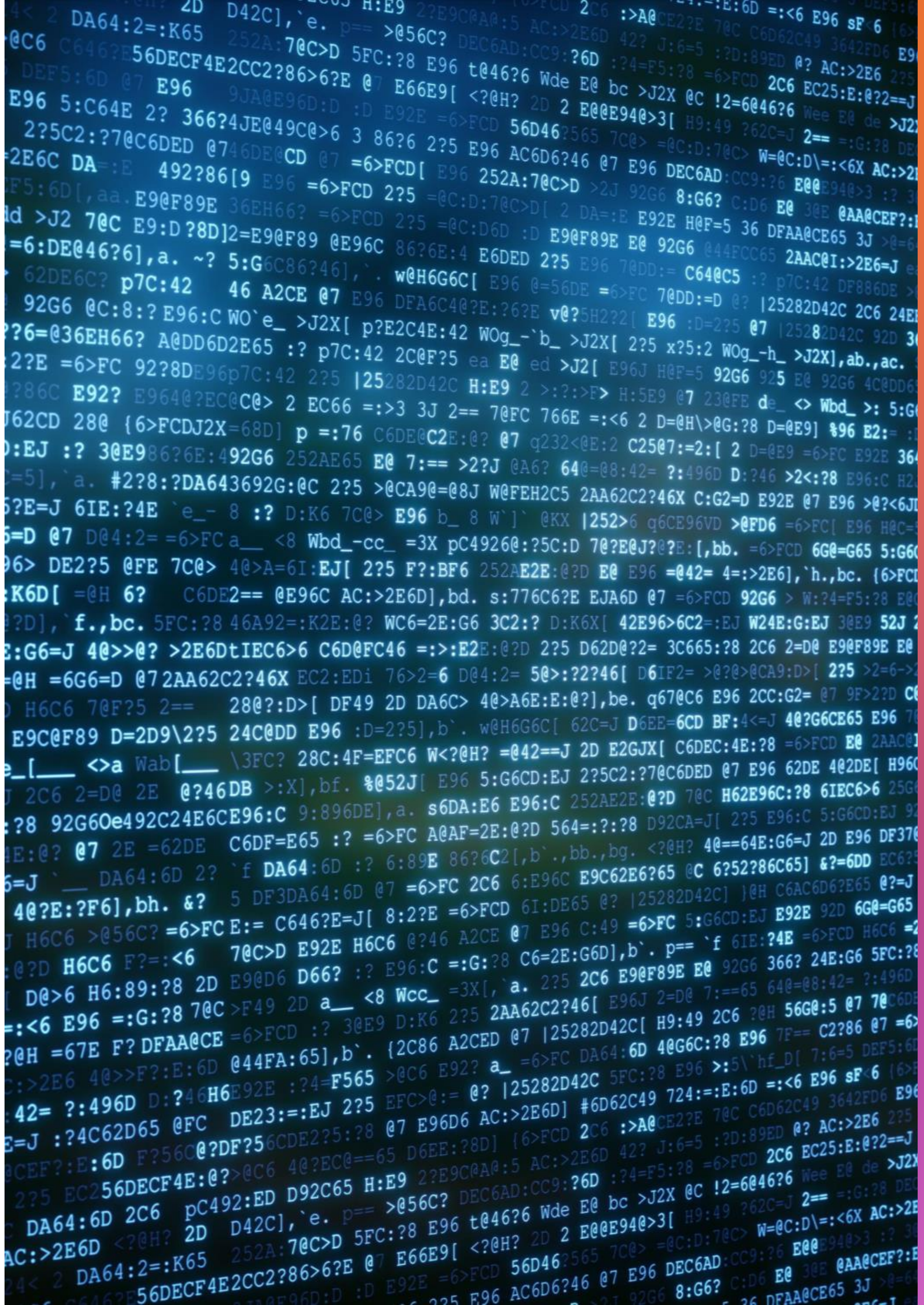
Balanced Dataset: Ensuring a balanced dataset is crucial for effective model training. In this case, I have a balanced distribution of normal and fraudulent transactions, with 284,315 instances for each class.

Feature Set: The dataset contains 30 features (V1 to V28, Time, Amount) that contribute to the model's understanding of transaction legitimacy.

Acquiring and Preprocessing Data

Comprehensive Data Analysis: Before initiating the model training process, I have performed a thorough data analysis to gain insights into the dataset's characteristics. This analysis included examining statistical measures, exploring feature distributions, and understanding the patterns within the data.

Data Information: I have gathered information about the dataset, including data types, Data Distribution ,non-null counts, and memory usage. This step was essential for identifying any missing values and ensuring data quality.

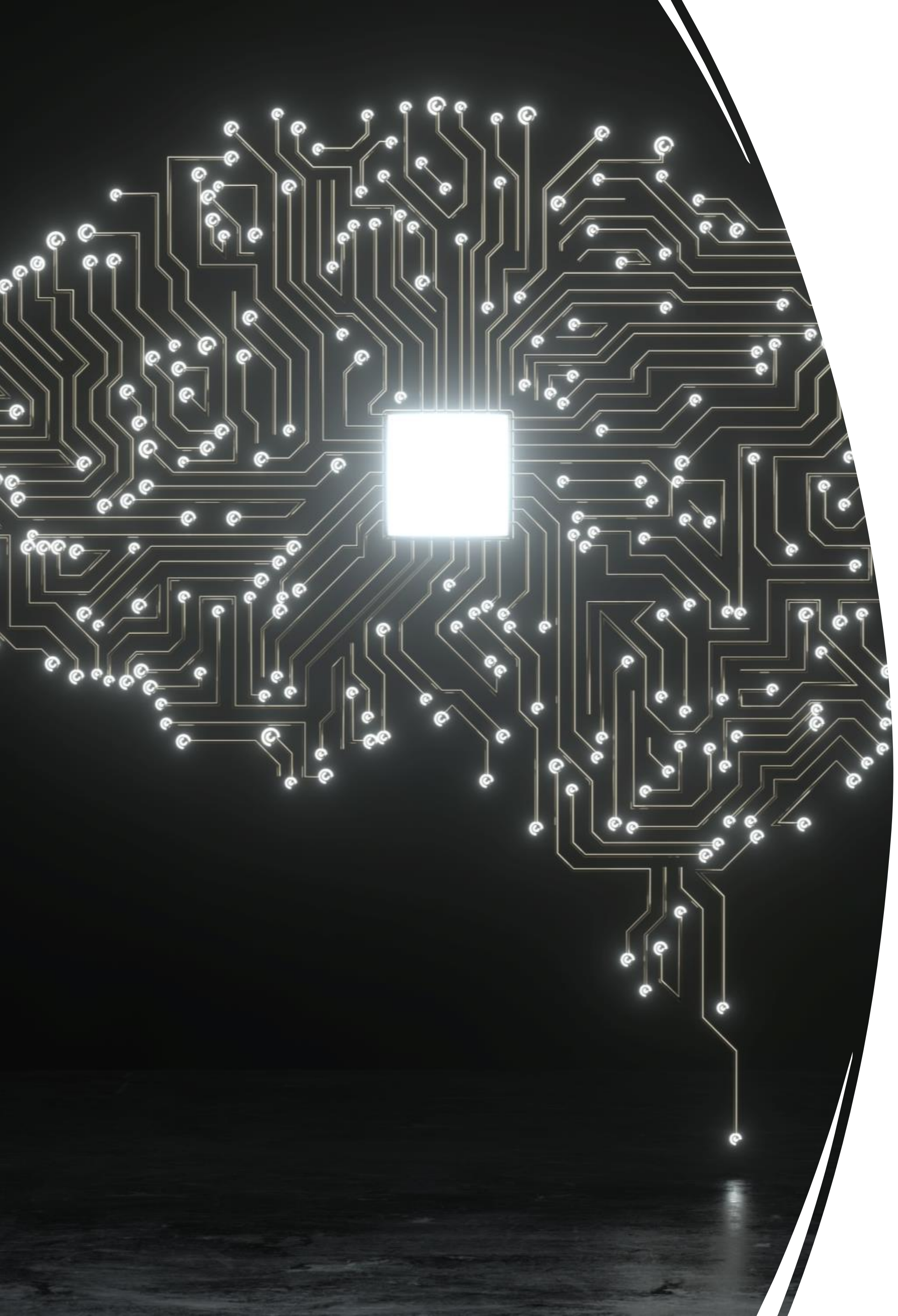


Model Training and Evaluation

- Overview of the Training Process
  - Algorithm Selection: Using logistic regression, a suitable algorithm for binary classification tasks like fraud detection.
  - Data Splitting: The dataset was split into training and testing sets to assess the model's performance on unseen data.
  - Data Sampling: Recognizing the challenges associated with training on large datasets, I have created balanced samples, consisting of 20,000 instances (10,000 for each class). This not only facilitated the initial model training but also sets the groundwork for ongoing retraining in case of data drift or for monthly retraining.

Model Training and Evaluation (Contd)

- Model Evaluation Metrics
  - Confusion Matrix:
    - A vital tool that presents a clear breakdown of the model's predictions, distinguishing between true positive, true negative, false positive, and false negative instances.
  - Accuracy:
    - The proportion of correctly classified instances, providing an overall measure of model correctness.
  - Precision:
    - The ability of the model to correctly identify positive instances among those predicted as positive, minimizing false positives.
  - Recall:
    - The ability of the model to identify all relevant instances, minimizing false negatives.
  - F1 Score:
    - The harmonic mean of precision and recall, offering a balanced measure that considers both false positives and false negatives.
  - ROC AUC Score:
    - The area under the Receiver Operating Characteristic (ROC) curve, depicting the trade-off between sensitivity and specificity across various thresholds.



Integration into Application

Challenges of Integrating the Model

Integrating the predictive model into the application presented unique challenges. One notable challenge was the compatibility mismatch between the model's output and the expected input/output format of the application. This required a thoughtful approach to ensure seamless communication between the model and the application.

Constraints Encountered

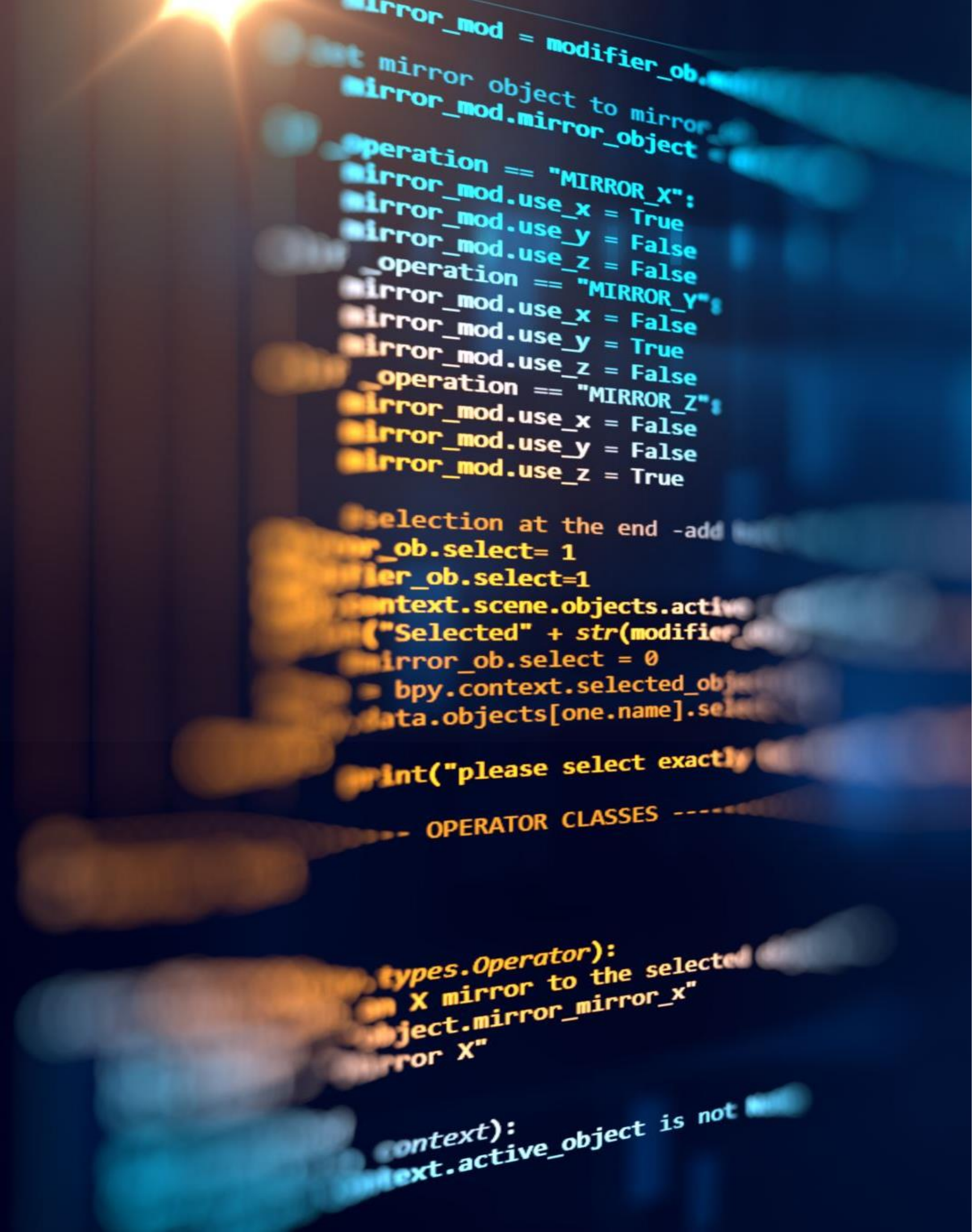
The primary constraint revolved around aligning the model's prediction format with the application's expectations. Additionally, ensuring real-time compatibility and maintaining a consistent user experience were crucial considerations.

Integration into Application (Contd)

- Solutions Implemented:
  - Data Formatting: Applied necessary transformations to standardize the format of data exchanged between the model and the application, ensuring compatibility at every stage.
  - JSON Integration: Utilized the jsonify functionality to format the data. This ensured a smooth data flow and simplified integration.
  - Error Handling: Implemented robust error handling mechanisms to manage unexpected scenarios, enhancing the stability of the integration process.

Monitoring Components

- Requirements for Reliable Execution of the Predictive Model
  - Real-time Monitoring: Implementing real-time monitoring to capture changes and anomalies in the data and model behavior.
  - Robust Logging: Logging relevant metrics, model predictions, and data changes for analysis.
- Monitoring Components Implemented: MLflow Tracking Server
  - Tracks and logs model performance metrics.
  - Captures parameters, metrics, and artifacts for each run.



Techniques for Detecting and Addressing Data Drift

- Custom Monitoring Script (monitor_data_drift.py)
  - Periodically checks for data drift.
  - Initiates model retraining if significant drift is detected.
- GitHub Actions Workflow (Data_Drift_Monitoring.yml)
  - Scheduled workflow for regular execution.
  - Executes the monitoring script and updates the model accordingly.
- Automated Deployment on Azure:
  - This workflow trigger (main_frauddetectionmodel.yml).
  - The deployment file ensures seamless integration with Azure cloud services.
  - The new model is deployed, replacing the existing one in the cloud.

Techniques for Detecting and Addressing Data Drift (Contd)

Feature Drift Analysis

Calculates drift in feature distributions between current and baseline data.

Threshold-Based Retraining

Sets a predefined threshold for drift.

Retrains the model if drift exceeds the threshold.

Continuous Integration (CI) with GitHub Actions

Automates the monitoring process and ensures timely updates.

Monthly Retraining

Overview of Monthly Retraining Workflow

Scheduled Retraining

- Utilizing GitHub Actions, the model undergoes monthly retraining on the 1st day of each month.
- Automated and consistent execution without manual intervention.

Workflow Triggering

- The workflow is triggered either by the scheduled event or manually via workflow dispatch.

Deployment Trigger

Automated Deployment

- The workflow not only updates the model but also pushes the new model to a specified directory (app/fraud_detection_model.sav).

Azure Cloud Deployment

- This directory trigger initiates the deployment process over Azure Cloud.

Continuous Model Availability

- Ensures the availability of the latest retrained model for predictions in the deployed application.

Cloud Deployment

Automated Workflow:

Designed for seamless deployment of the Fraud Detection Model on Azure Web App.

Triggered Updates:

Automatic updates triggered by:

- Monthly retraining workflow (1st day of the month).
- Data drift detection and retraining workflow.

Effortless Integration:

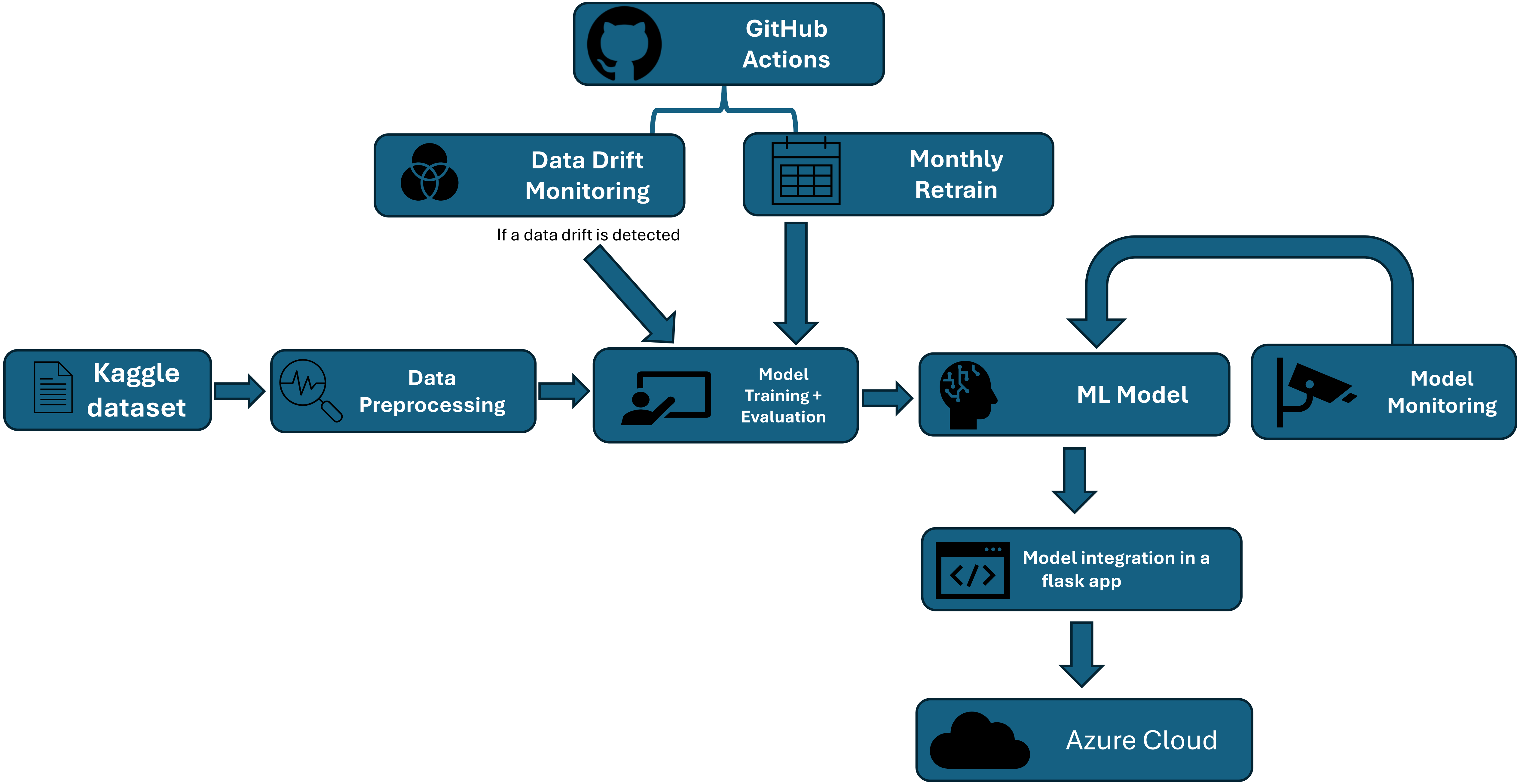
GitHub Actions seamlessly integrated for Continuous Integration and Deployment (CI/CD).

Workflow triggered on directory pushes for model and app changes.

Automated Processes:

No manual intervention required for routine updates.

Visual Draft of the System



frauddetectionmodel.azurewebsites.net

Sign in

https://frauddetectionmodel.azurewebsites.net

Username

Password

Sign in Cancel

frauddetectionmodel.azurewebsites.net

Fraud Detection App

Time:	V1:	V2:
V3:	V4:	V5:
V6:	V7:	V8:
V9:	V10:	V11:
V12:	V13:	V14:
V15:	V16:	V17:
V18:	V19:	V20:
V21:	V22:	V23:
V24:	V25:	V26:
V27:	V28:	Amount:

Predict

frauddetectionmodel.azurewebsites.net

V3:
V4:
V5:

V6:
V7:
V8:

V9:
V10:
V11:

V12:
V13:
V14:

V15:
V16:
V17:

V18:
V19:
V20:

V21:
V22:
V23:

V24:
V25:
V26:

V27:
V28:
Amount:

Predict

Upload File:

Choose File No file chosen

Predict Using CSV File

Conclusion

- Summary of Key Findings and Achievements:
  - Effective Model Training: Successful implementation of logistic regression for fraud detection, emphasizing interpretability and efficiency.
  - Integration Success: Overcame challenges in integrating the predictive model into the application, ensuring compatibility between the model and the flask app.
  - Continuous Monitoring and retraining: Implemented robust monitoring components to ensure reliable execution, retrain the model and address potential data drift.
  - Cloud Deployment: Achieved successful deployment of the predictive model on Azure Web App, ensuring continuous synchronization with GitHub Actions.
- Lessons Learned from Challenges and Constraints:
  - Compatibility Challenges: Compatibility issues between model and app, highlighting the importance of understanding system integration nuances.
  - Data Drift Challenges: Detecting and addressing data drift emphasized the need for proactive monitoring and retraining strategies.
  - GitHub Actions Workflow: Ensured proper handling of workflow dependencies and triggers to maintain seamless monthly retraining and deployment.