VCC PROJECT

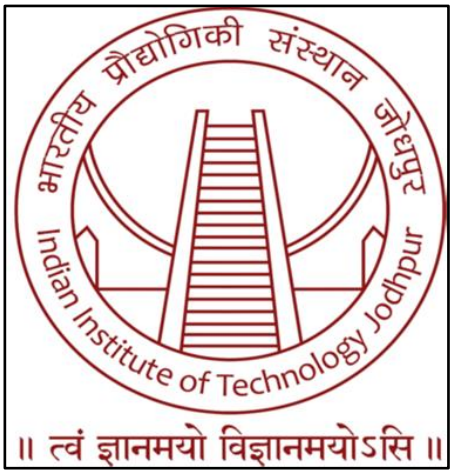
Harnessing Cloud Computing for Fraud Detection in Online Payments

Team Members:

1. Shubhankit Dutt (G23AI2070)
2. Nihar Dave (G23AI2097)
3. Priyank Bhardwaj (G23AI2038)

Under the Supervision of

Prof. Sumit Kalra



PG DIPLOMA in DATA ENGINEERING

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

INDIAN INSTITUTE OF TECHNOLOGY JODHPUR

SEP, 2024

1. Introduction

1.1. Project Overview

Objective: The project aims to develop a machine learning-based fraud detection system capable of accurately identifying fraudulent online transactions while minimizing false positives. The system will be accessible as a web service to support real-time transaction monitoring. The project includes building a fraud detection model using various machine learning algorithms, deploying it on Google Cloud, and creating a user-friendly interface for accessing predictions.

1.2. Background

As e-commerce grows, the threat of online payment fraud has escalated, leading to financial losses and reduced consumer trust. Advanced detection systems using machine learning are necessary to effectively identify fraudulent activities and protect financial transactions. This project leverages historical transaction data to train models that generalize well to new, unseen data.

2. Requirements

2.1. Functional Requirements

- Data Ingestion: The system should accept transaction data in CSV format, including features like transaction amount, type, and account balances.
- **Fraud Detection:** The model should apply various classification algorithms to detect fraudulent transactions.
- Model Training: Ability to retrain the model periodically with new transaction data to adapt to evolving fraud patterns.

2.2. Non-Functional Requirements

- Scalability: The system should scale horizontally to accommodate increased traffic.
- Availability: 99.9% uptime, leveraging GCP's managed services for high availability.

3. Architecture and Design

3.1. System Architecture

Google Cloud Platform (GCP): The application will use GCP for deployment, leveraging Google App Engine (GAE) for hosting.

Components:

- Frontend: A user-friendly web interface**.**

- Backend: Python/Flask-based API providing access to the fraud detection model.

- Model: Machine learning model i.e. Random Forest implemented in Python using scikit-learn.

- **Monitoring:** GCP logging and monitoring services.

3.2. Database Design

Google Cloud Storage: For storing transaction data as CSV files, which the model will periodically ingest for retraining.

4. Workflow and Process

1. User uploads transaction data via UI.
2. Data is stored in Google Cloud Storage.
3. Fraud detection model is trained using transaction features.
4. ML model is used to classify transactions for predictions.

5. Technology Stack

- **Modeling:** Python, scikit-learn for building classification models.

- Application Development: Python with Flask and html for UI.

- Deployment: Google App Engine for scalable deployment.

- Monitoring: GCP logging and performance monitoring.

6. Security Considerations

- Data Encryption: All data (in transit and at rest) will be encrypted using GCP's default encryption standards.

- Authentication: Roles and permissions managed via Google Identity and Access Management (IAM) for users and service accounts.
- Access Control: Restrict access to the GCP environment through IAM roles.

7. Performance Considerations

- Model Latency: The fraud detection model will be optimized to respond to classification requests within 100 ms for up to 1000 concurrent requests.

- Load Balancing: Google App Engine's auto-scaling will handle traffic spikes, ensuring that the App remains performant.

- Caching: Consider using in-memory caching (e.g., Redis) to store recent classifications.

8. Testing and Validation

8.1. Unit Testing
You may implement unit tests for each part of the model functions using pytest.

8.2. Integration Testing
Test end-to-end integration by simulating requests to the app and checking the model's responses. Testing to be performed on localhost first and check whether the model is running correctly.

8.3. User Acceptance Testing (UAT)
Allow internal users (fraud analysts) to test the system and provide feedback on prediction accuracy.

9. Deployment Strategy

- Development Environment: Build the model locally and test on your local machine by creating a virtual environment.

- Staging: Deploy to a GCP staging environment for UAT.

- Production: Deploy the model on Google App Engine, enabling auto-scaling and ensuring proper logging and monitoring.

10. Risks and Mitigation

- Risk: Model may become outdated as fraud patterns evolve.
- Mitigation: Implement periodic retraining of the fraud detection model using new data from Cloud Storage.

- **Risk:** Scalability issues during traffic spikes.
- Mitigation: Use App Engine's auto-scaling feature and monitor performance via Google cloud logging and monitoring services.

11. Timeline

Phase 1: Initial Development

- Data preparation, feature engineering, model training, and initial testing.

Phase 2: API and GCP Integration

- Model development, testing, and deployment on Google App Engine.

Phase 3: Final Testing and Go-Live

- User Acceptance Testing (UAT), performance optimization, and go-live deployment.

Thank You