Anomaly Detection in Financial Transactions

Deployment phase.

Group Members

1. Ayana Damtew Tajebe
2. Beamlak Dejene
3. Anwar Mohammed Koji
4. Anwar Gashaw Yimam
5. Amen Zelealem Tadese

Machine Learning Project Documentation

Deployment

---

1. Overview

This phase focuses on deploying the trained machine learning models for detecting anomalies in transaction data into a production-ready environment. The deployment involves serializing the models, serving them through a RESTful API, securing the service, and setting up monitoring to ensure consistent performance. The service is hosted on Render and accessible via a public endpoint for real-time anomaly detection.

---

2. Model Serialization

The trained models—**Isolation Forest** and **Autoencoder**—are serialized to allow efficient storage and reloading without retraining:

- **Isolation Forest:** Serialized using `joblib` and saved as `isolation_forest_model.joblib`.
- **Autoencoder:** Saved using Keras's `save` method as `autoencoder_model.h5`.

Key considerations:

- **File size optimization:** Models are compressed to reduce storage overhead.
- **Compatibility:** Serialization formats chosen are compatible with the deployment environment.
- **Versioning:** Model files are version-controlled to track updates and enable rollback.

---

3. Model Serving

The serialized models are served via a FastAPI-based RESTful service hosted on Render:

- **Hosting platform:** Render (`https://anomalydetection-lgh3.onrender.com`)
- **Service:** Loads models on startup and serves prediction requests with low latency.
- **Endpoints:**
    - `/predict_isolation_forest`
    - `/predict_autoencoder`

This setup supports scalable and reliable real-time anomaly detection for transaction data.

---

4. API Integration

The deployed API allows clients to send transaction data and receive anomaly predictions.

- **Base URL:** `https://anomalydetection-lgh3.onrender.com`
- **Endpoints:**
  - **POST `/predict_isolation_forest`**
    Accepts transaction features as JSON and returns anomaly prediction from the Isolation Forest model.
  - **POST `/predict_autoencoder`**
    Accepts transaction features as JSON and returns anomaly prediction from the Autoencoder model.
- **Input Format:**
  JSON object with transaction features, e.g.:

  ```
  json
  CopyEdit
  {
    "Transaction_Amount": 150.75,
    "Average_Transaction_Amount": 120.00,
    "Frequency_of_Transactions": 3
  }
  ```

- **Response Format:**
  JSON indicating anomaly status:

  ```
  json
  CopyEdit
  {
    "is_anomaly": false
  }
  ```

- **Input validation** ensures data correctness and guards against malformed requests.

---

5. Security Considerations

To protect the API and sensitive data, several security practices are implemented:

- **Authentication:** OAuth 2.0 with JWT tokens protects endpoints from unauthorized access.
- **Authorization:** Role-based controls ensure only permitted users access specific features.
- **Encryption:** HTTPS with TLS encrypts data in transit.
- **Input Validation:** Strict validation prevents injection attacks and enforces data integrity.

6. Monitoring and Logging

Ongoing monitoring ensures high availability and performance:

- **Metrics monitored:** API response time, request rates, error rates, and prediction latency.
- **Logging:** Requests and responses are logged with timestamps and severity levels (INFO, WARNING, ERROR).
- **Tools:** Prometheus collects metrics; Grafana provides dashboards for real-time visualization.
- **Alerts:** Configured to notify via email/SMS on high error rates or degraded performance.
- **Health checks:** Periodic checks verify API uptime and responsiveness.

References

- GitHub Repository: Anomaly-Transaction-Detection
- Deployed API: https://anomalydetection-lgh3.onrender.com/
