Real-time Lead Scoring: Production-Ready Serving Infrastructure

This repo delivers a **production-quality** reference implementation for a real-time lead scoring system using **API Gateway** and **AWS Lambda**, model deployed on **Amazon SageMaker** endpoint with scalable instances and **Terraform** for IaC (not using due to resources constraint). It is designed to handle **~300 RPS** with **p99 < 1s** under typical payload sizes, and demonstrates best practices for **security, observability, CI/CD, testing, and MLOps**.

Assumption

- Assumption is that our model is ready to production hence created a dummy model in local
- Data pipeline for all the sources is already there from which we are getting input features for the xgboost model

---

Operations

Deployment

Local Setup -- Need aws credential and configure in Local

```
#Run a make file for upload the model from model
make
```

Github Actions

PROF

Through github actions deployment can be triggered for `main` branch PR

- Build stage
- Terraform is in one stage but will be implemented in future(Right now resources like s3 bucket, lambda function and api gateway done manually once).
- Security check for PII and other sensitive information
- Unit test
- Code review through ruff
- Deploy the model and create the sagemaker endpoint

Load Testing with Locust

- We have included a load testing setup using Locust.
- This allows you to simulate concurrent requests and validate system performance under load.

Setup

```
cd load_tests
pip install -r requirements.txt
locust -f load_tests/locustfile.py --host <API Gateway Endpoint>
```

CI/CD

- **GitHub Actions**
  - Lint (`ruff`), type-check (`mypy`), unit tests (`pytest`, coverage)
  - Security: `bandit` (py)
  - Deployment and creation of sagemaker endpoint for xgboost model
  - Implement the lambda function
  - **staging** on main branch; manual approval for **prod**

Endpoints (Once deployment completed sagemaker endpoint deployed)

- `POST /score` — returns a score 1–5 and latency metadata

Sample request: (list of numpy arrays each of 50 feature)

```
[
  [0.0,0.4,....,0.3],
  [0.0,0.4,....,0.3]
]
```

PROF

High-Level Architecture

- **Ingress**: AWS API Gateway + AWS Lambda (maintable,scalable and secure)
- **Compute**: Sagemaker (Autoscale Instances)
- **Model Inference**:
  - Option A (default demo): In-process mock XGBoost-compatible scorer
  - Option B (prod): Call a **SageMaker real-time endpoint** (recommended for heavy models)
- **Data Lake Writes**: Results are pushed asynchronously to **Kinesis Firehose -> S3** (parquet) with partitioning (Future Implementation)
- **Features**: 50 features accepted; schema validated with
- **Observability**:
  - **Cloudwatch** log ingested for execution
  - **SageMaker Model Monitor** (template) for data/quality drift and concept drift(Future Implementation)
- **Security**:

- Secrets in **Github Repo**

Scope & Cost Decisions

- **Implemented**: aws runnable service, API gateway+ AWS Lambda, sagemaker model, deploy endpoint for sagemaker, Cloudwatch log for requests, CI security gates, Unit test cases,
- **Documented templates**: SageMaker endpoint + Model Monitor
- **Deferred**: DataPipeline for input from different sources, Feature Engineering, Snowflake, end-to-end Lakehouse table creation, Terraform,full Grafana stack (CloudWatch metrics suffice for demo), Logs for drift detection, alarm from cloudwatch for thoughput or latency

Future Improvements

- Advanced drift monitors and bias metrics with alerts to Slack (SNS)
- Terraform for various cloud like gcp and on-prem
- Automate the complete MLOps life cycle - Auto Retraining, Auto Feature Engineering, Beta Testing,