END-TO-END ML PIPELINE WITH DEPLOYMENT

1. Project Overview

This project focuses on building an end-to-end **Machine Learning Pipeline** capable of fully automated data ingestion, feature engineering, model training, hyperparameter tuning, evaluation, and deployment.
The system is designed to be scalable, modular, and production-ready, with CI/CD-driven model retraining and automatic artifact versioning.

The pipeline supports multiple ML algorithms—including **XGBoost, LightGBM, and Random Forest**—and selects the best-performing model based on validation metrics.
The final model is deployed as a **REST API** using FastAPI/Flask, enabling real-time predictions.

---

2. Problem Statement

Organizations often struggle with ML models that:

- Break due to data drift
- Are trained manually without automation
- Lack reproducibility
- Cannot be deployed easily
- Perform inconsistently across datasets

This project solves these challenges by building a **production-grade ML workflow** that automates the entire lifecycle from data to deployment.

---

3. Objectives

- Build a modular ML pipeline with reusable components
- Automate ingestion, cleaning, and feature extraction
- Integrate model training with hyperparameter tuning

- Support multiple algorithms with automated model selection
- Deploy the best model as an inference API
- Enable reproducible experiments using proper artifact logging
- Add CI/CD automation for scheduled retraining

---

4. Tech Stack

Languages

- Python

Frameworks & Libraries

- pandas, NumPy
- scikit-learn
- XGBoost, LightGBM
- joblib, pickle
- FastAPI / Flask
- Docker
- GitHub Actions (CI/CD)

Tools

- Git
- DockerHub
- VS Code
- Postman for API testing

5. System Architecture

```
┌──────────────────────┐
│   Raw CSV Input      │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│   Data Ingestion     │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Data Cleaning + FE   │
│ (scaling, encoding)  │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Train/Val/Test Split │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Multi-Model Training │
│ XGB / LGBM / RF      │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Best Model Selection │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Artifact Saving      │
│ (scaler + model)     │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Deployment API       │
└──────────────────────┘
```

6. Data Pipeline

6.1 Ingestion

- **Loaded raw data from CSV files**
- **Validated schema (rows, missing values, data types)**
- **Added timestamp-based indexing**

6.2 Cleaning

- **Imputed missing values using median strategy**
- **Removed outliers using IQR**
- **Normalized skewed distributions**
- **Checked data leakage**

6.3 Feature Engineering

- **Scaled numerical features with StandardScaler**
- **Encoded categorical variables**
- **Generated interaction features**
- **Removed low-variance features**
- **Applied correlation-based feature selection**

6.4 Dataset Splitting

A 70–10–20 split was used:

- **70% training**
- **10% validation**
- **20% testing**

---

7. Model Development

Models Used

- **Random Forest Classifier**
- **XGBoost Classifier**
- **LightGBM Classifier**

Each model underwent:

- **Hyperparameter tuning via GridSearchCV**
- **Cross-validation**
- **Class imbalance handling**
- **Performance scoring on validation data**

Model Selection Criteria

The best model was chosen using:

- **Accuracy**
- **F1-score**
- **ROC-AUC**
- **Validation loss**

The pipeline automatically logs:

- **Best model**
- **Metrics**
- **Hyperparameters**

---

8. Evaluation Metrics

Model	Accuracy	F1 Score	AUC
Random Forest	0.91	0.89	0.93
XGBoost	0.92	0.90	0.94
LightGBM	0.93	0.92	0.96

LightGBM was selected as the final deployed model.

---

9. Deployment

9.1 API Development

A FastAPI/Flask service was created to serve real-time predictions.

Routes:

- `/predict` — returns prediction + probabilities
- `/health` — health check endpoint

9.2 Dockerization

- Containerized app using Docker
- Pushed to DockerHub
- Prepared image for cloud deployment

9.3 CI/CD (Retraining + Deployment)

Using GitHub Actions:

- On data update → trigger retraining
- On code update → run unit tests
- Auto-build & deploy Docker image

- **Notify via GitHub checks**

---

10. Challenges & Solutions

Challenge 1: Data Drift Across Time

Solution: Added drift detection module + scheduled retraining.

Challenge 2: Large Feature Space

Solution: Automated feature importance pruning based on SHAP + permutation importance.

Challenge 3: Maintaining Reproducibility

Solution:

- **Versioning models**
- **Logging dependencies in requirements.txt**
- **Saving scaler + encoder for future predictions**

---

11. Conclusion

The project successfully delivers an automated, production-ready ML workflow with:

- **End-to-end pipeline**
- **Multiple model support**
- **Automated selection**
- **API deployment**
- **Drift detection and retraining**

This system can be applied to fintech, healthcare, retail forecasting, fraud detection, and other industry ML use cases.