Architecture

Restaurant Rating Prediction

Contents

1. Introduction

The Restaurant Rating Prediction Tool is a machine learning-based web application that predicts restaurant ratings based on key features such as restaurant type, location, online ordering availability, and user votes. The system follows a structured pipeline for data processing, model training, and deployment using Flask, AWS, Cassandra, and GitHub Actions.

This document outlines the system's architecture, including data ingestion, preprocessing, model training, deployment, and automation.

2. System Architecture Overview

2.1 High-Level Workflow

1. **Data Collection & Storage:** Data is ingested from Kaggle and stored in Cassandra.

2. **Data Processing:** Cleaning, handling missing values, and transforming data for model training.

3. **Model Training & Evaluation:** Training an ML model (e.g., Random Forest) for rating prediction.

4. **Deployment:** Flask-based web interface deployed on AWS with CI/CD using GitHub Actions and AWS CodePipeline.

3. Architecture Breakdown

3.1 Data Pipeline

Step 1: Data Ingestion

- The dataset is sourced from Kaggle and downloaded into the system.
- Data is read into Pandas for initial exploration.

Step 2: Data Cleaning

- Handle null values by imputing or dropping missing data.
- Ensure consistent formatting of categorical variables.

Step 3: Upload to Cassandra

- A Cassandra NoSQL database is used for scalable and efficient data storage.
- Data is inserted into structured tables for easy retrieval.

Step 4: Download from Cassandra

- Preprocessed data is retrieved from Cassandra for further processing.

Step 5: Data Preprocessing

- Handling outliers to improve model performance.
- Dropping unnecessary columns to keep only relevant features.

Step 6: Data Transformation

- **Label Encoding** for categorical variables.
- **Standard Scaling** for numerical features to normalize data.

Step 7: Model Training

- Train a Random Forest or XGBoost model for rating prediction.
- Evaluate the model using metrics like R^2 score.

3.2 Prediction Pipeline

Frontend & Backend Integration

- The trained model is integrated into a Flask API to handle user input and return predictions.
- A web interface (HTML + Flask) allows users to enter restaurant details and receive a predicted rating.

Backend Workflow

1. **User Input:** The frontend form collects restaurant details.
2. **API Processing:** The input is sent to Flask, which applies transformations and feeds it to the model.
3. **Model Prediction:** The trained model predicts the restaurant rating.
4. **Output Display:** The predicted rating is shown on the frontend.

3.3 Deployment & CI/CD

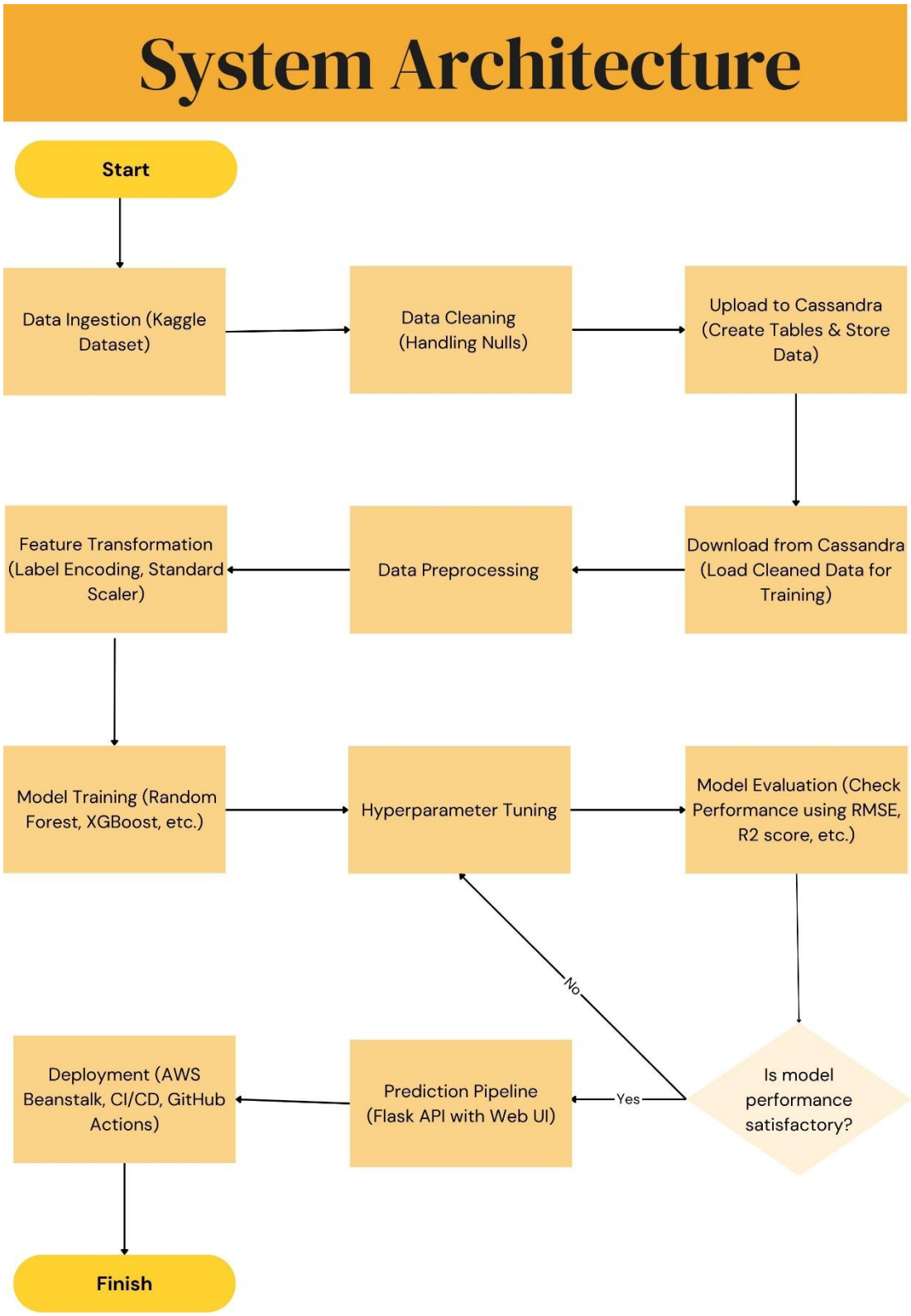
Deployment on AWS

- The Flask application is deployed on an AWS Elastic Beanstalk.
- S3 Buckets may be used for storing model artifacts.

CI/CD with GitHub Actions & AWS CodePipeline

- **GitHub Actions:** Automates testing and deployment.
- **AWS CodePipeline:** Manages continuous deployment, ensuring the latest version is deployed.

4. Architecture Diagram



System Architecture Breakdown	
Process Stage	Description
Data Ingestion (Kaggle Dataset)	Data is sourced from Kaggle and stored in the system.
Data Cleaning (Handling Nulls)	Missing values are handled.
Upload to Cassandra (Create Tables & Store Data)	Data is stored in Cassandra after structuring.
Download from Cassandra (Load Cleaned Data for Training)	Cleaned data is retrieved for model training.
Data Preprocessing	Irrelevant features and outliers are removed.
Feature Transformation (Label Encoding, Standard Scaler)	Converts categorical data into numerical and normalizes it.
Model Training (Random Forest, XGBoost, etc.)	ML models are trained using transformed data.
Hyperparameter Tuning	Optimizing model parameters for better performance.
Model Evaluation (Check Performance using RMSE, R2 score, etc.)	Measuring model accuracy and error using metrics.
Prediction Pipeline (Flask API with Web UI)	• Flask API handles user input. • The trained model predicts restaurant ratings. • The Web UI displays the predicted rating.
Deployment (AWS Beanstalk, CI/CD, GitHub Actions)	The model is deployed using Flask on AWS Beanstalk with automation.