07.08.2024

Plant Recognition

Project Specification (MLOPs)

Inhalt [tribute to Germany]

Team: Luigi Menale | Arif Haidari | Alex Tavkhelidze

mentor: Sebastien

DATASCIENTEST

1. Context and Objectives

The agricultural industry faces significant challenges in monitoring plant health and diagnosing diseases promptly. Early detection of diseases is crucial for preventing widespread crop damage, ensuring food security, and optimizing agricultural output. This project aims to develop an AI-driven solution that can detect plant species and diagnose diseases from images, providing actionable insights to farmers and agronomists.

1.1. Potential Stakeholders:

-i- App Sponsor: The project is sponsored by a hypothetical agricultural technology company, AgriTech Solutions, aiming to enhance precision agriculture through advanced AI tools.

-ii- Users: The primary users include farmers, agronomists, agricultural consultants, and researchers who need to monitor crop health, diagnose plant diseases, and make informed decisions based on the health status of plants.

-iii- Application Administrators: The system will be managed by IT specialists from AgriTech Solutions, responsible for maintaining the application's infrastructure and ensuring continuous operation.

1.2. Integration Context:

The primary interface will be a REST API, enabling easy integration with existing software solutions. Additionally, a minimalistic graphical interface will be available for system administrators to monitor application health and manage user permissions.

2. Model

The system employs a Convolutional Neural Network (CNN) for image classification tasks. The model processes images to:

- Identify the plant species.
- Determine if the plant is healthy or diseased.
- If diseased, classify the specific disease affecting the plant.

2.1. Model Performance:

-i- Accuracy: The CNN achieves an accuracy of approximately 97% in species identification and disease detection.

-ii- Robustness: The model predicts well enough on the homogenous data (on the test dataset which are mostly one single leaf image) but on the heterogeneous dataset the accuracy is low.

-iii- Training Time: Training took approximately 10 hours on Standard Laptop.

-iv- Prediction Time: The prediction time depends on the processing power and also on the platform which is deployed.

2.2. Model Metrics:

-i- F1 Score to balance precision and recall, providing a single metric for model

-ii- Accuracy for assessing the proportion of correct predictions.

3. Database

The project utilized static image datasets sourced from public repositories and field images. The data included labeled images of various plant species and associated diseases, with a focus on leaf imagery.

3.1 Data Management and Ingestion:

For the project, data remains static; however, in a real-world scenario, the system would integrate dynamic data ingestion pipelines. These pipelines would:

- Allow uploading new images from field surveys or crowd-sourced platforms.
- Include automated data cleaning and labeling workflows.
- Support continuous model updates with the latest data to ensure the system's accuracy and relevance.

4. API

4.1 API Endpoints:

-o- POST /predict: Receives an image and returns the detected plant species and disease diagnosis.

-i- POST /train: Allows authorized users to retrain the model with new data inputs.

-ii- POST /upload-data: Endpoint for administrators to upload new datasets for model training.

-ii- GET /status: Provides real-time status information on the system's operational health, including model performance metrics and API uptime.

4.2 Authentication and Security:

The API employs token-based authentication to secure access, ensuring that only authorized users can interact with sensitive endpoints like model retraining and data uploads. This setup prevents unauthorized access and ensures data integrity.

5. Testing & Monitoring

5.1 Unit Testing:

-o- Model Training: Unit tests verify that the model can be trained without errors and that it performs as expected on a validation dataset.

-i- Prediction Testing: Ensures that the model accurately predicts species and diseases from provided images.

-ii- API Endpoint Testing: Tests to confirm that API endpoints respond correctly to valid and invalid requests, ensuring robust error handling.

-iii- Data Ingestion Testing: Validates the data ingestion process, checking for proper data formatting and integration into the training pipeline.

5.2 Monitoring:

-o- Performance Monitoring: Continuous tracking of the model's accuracy and precision using a test set and recent data samples. The system includes dashboards for real-time performance visualization.

-i- Retraining Schedule: The model will be retrained periodically or when a significant drop in performance is detected. Retraining can also be triggered manually through the admin interface.

-ii- Alerts and Notifications: The system sends alerts to administrators if model performance drops below a predefined threshold or if critical errors occur. This includes notifications for issues like data ingestion failures or API downtime.

6. Implementation

The implementation scheme for the Plant and Disease Detection System integrates multiple components and tools to ensure seamless operation, scalability, and maintainability. This section provides a detailed overview of the architecture, illustrating how different elements interact and the specific tools used in the project.

It is important to note that the tools and components outlined below are the intended or theoretical choices considered at the project's inception. However, as the project progresses, other tools or components may come into consideration due to various factors such as performance requirements, integration challenges, cost considerations, or emerging technologies. Flexibility in tool selection allows the project to adapt and incorporate the most effective solutions available, ensuring optimal performance and sustainability.

6.1 Data Ingestion and Preprocessing:

Components Involved:

- Data Source: Static datasets and dynamic data inputs (e.g., new images uploaded by users).
- Data Ingestion Pipeline: This pipeline handles the collection, validation, and transformation of data before it is fed into the model.

Tools:

- Airflow: Orchestrates the data ingestion pipeline, scheduling tasks for data collection, preprocessing, and storage. It ensures that data is consistently processed and loaded into the system at the correct intervals.
- Containers encapsulate the preprocessing scripts, ensuring that they run consistently across different environments. This includes image resizing, normalization, and augmentation processes.
- Kubernetes: Manages the deployment of Docker containers, scaling them based on the workload and ensuring high availability.

6.2 Model Training Deployment:

Components Involved:

- Machine Learning Model: A Convolutional Neural Network (CNN) used for image classification.
- Model Training Pipeline: Involves the training, validation, and evaluation of the model.

Tools:

- MLflow: Used for experiment tracking, model versioning, and registering models. It allows data scientists to track different experiments, compare results, and manage model lifecycle.
- GitHub Actions: Automates the CI/CD pipeline, including model training, testing, and deployment. Upon committing code changes, GitHub Actions triggers workflows that build Docker images, run unit tests, and deploy the updated model.
- Kubernetes: Hosts the trained model in a scalable environment, using Kubernetes pods to serve predictions. The cluster management capabilities of Kubernetes ensure that the model is always available and can handle varying loads.

6.3 API Gateway:

Components Involved:

- REST API: Serves as the interface for users and other systems to interact with the model and database.

Tools:

- FastAPI: A high-performance web framework for building APIs, it serves the endpoints for model inference, data upload, and system status checks. FastAPI's asynchronous capabilities ensure low latency and high throughput.
- Nginx: Acts as a reverse proxy and load balancer, distributing incoming requests across multiple instances of the FastAPI application. It also handles SSL termination, ensuring secure communication.

6.4 Monitoring and Logging:

Components Involved:

- Monitoring System: Tracks system health, performance metrics, and logs.
- Alerting System: Notifies administrators of critical issues.

Tools:

- Grafana: Visualizes real-time data on system performance, such as CPU usage, memory usage, API response times, and model accuracy metrics. Grafana dashboards provide insights into the system's operational state and help in making informed decisions.
- Elasticsearch, Logstash, and Kibana (ELK Stack): Aggregates and analyzes logs from all components. Logstash collects and processes log data, Elasticsearch stores it, and Kibana provides an interface for searching and visualizing the logs. This setup helps in troubleshooting and auditing.

6.5 Security and Access Control:

Components Involved:

- Authentication and Authorization: Ensures secure access to the system.

Tools:

- OAuth 2.0: Implements token-based authentication for API endpoints, ensuring that only authorized users can access sensitive functions like model retraining and data uploads.

6.4 Data Storage and Management:

Components Involved:

- Database: Stores raw data, processed data, and model predictions.

Tools:

- PostgreSQL: A relational database system used to manage structured data, such as metadata about images, user information, and system logs. It provides robust querying capabilities and supports complex transactions.

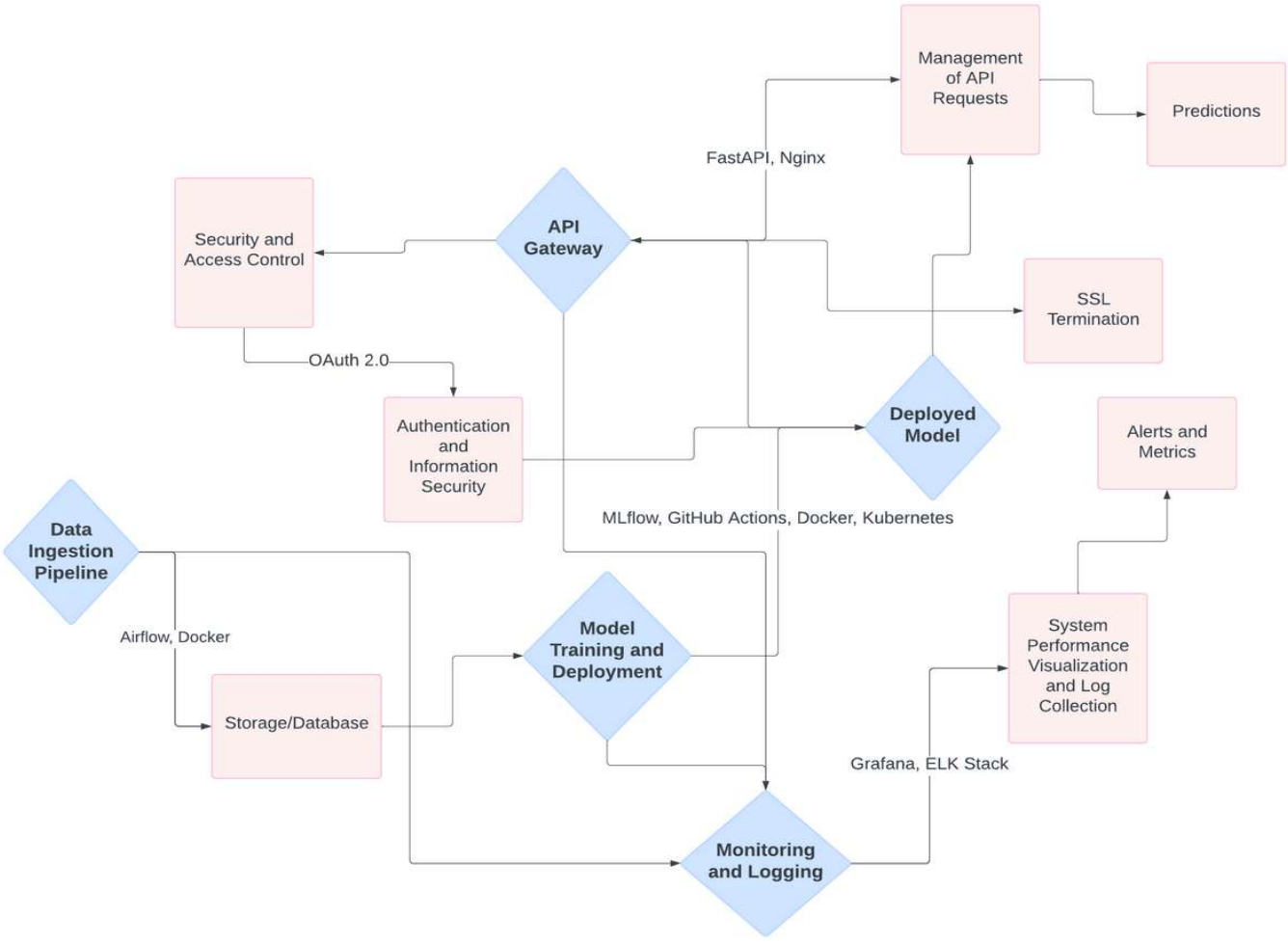

Figure 1: The summary diagram depicts the Plant and Disease Detection System's architecture, showing data ingestion, preprocessing, model training, deployment, API management, monitoring, and security components.

7. References:

1 LeCun, Y., Bengio, Y., & Hinton, G. (2015). "Deep learning." Nature, 521(7553), 436-444.
https://www.nature.com/articles/nature14539
Accessed on 06.08.2024

2 Deep learning in agriculture: A survey
https://www.sciencedirect.com/science/article/abs/pii/S0168169917308803
Accessed on 07,08.2024

3 Wheat yield prediction using machine learning and advanced sensing techniques
https://www.sciencedirect.com/science/article/abs/pii/S0168169915003671
Accessed on 07,08.2024

4 Apache Airflow Documentation
https://airflow.apache.org/docs/
Accessed on 05,08.2024

5 Grafana Documentation
https://grafana.com/docs/grafana/latest/
Accessed on 05,08.2024

6 MLflow Documentation
https://mlflow.org/docs/latest/index.html
Accessed on 05,08.2024