Wildfire Detection with NASA FIRMS Data & Automated MLOps Pipeline on Google Cloud Platform

Abstract

Wildfires represent a critical global environmental challenge, demanding innovative technological solutions for early detection and rapid response. This capstone project addresses this imperative through an automated machine learning system that predicts wildfire probability and intensity using satellite data analytics. Leveraging NASA's FIRMS dataset and Google Cloud Platform, we engineered a production-grade MLOps pipeline that continuously improves prediction capabilities without manual intervention.

Our solution features two complementary deep learning models: a multi-class classifier predicting fire detection confidence (nominal/low/high), and a regression model estimating Fire Radiative Power (FRP) to quantify wildfire severity. These TensorFlow models were trained on 21 million global fire observations, employing stratified sampling and SMOTE techniques to address class imbalance while maintaining geographical representation.

Validation results demonstrate 89.2% classification accuracy and 84.7% FRP prediction efficiency against historical fire events. The system successfully navigated challenges including artifact packaging, permission management, and computational constraints through infrastructure-as-code principles.

This implementation establishes a scalable template for environmental monitoring systems, providing agencies with continuously evolving wildfire intelligence. Future enhancements will incorporate scheduled retraining, global endpoint distribution, and satellite imagery analysis to further strengthen prediction capabilities.

Table of Contents

1. Introduction

Wildfires have become an increasingly urgent global problem, threatening ecosystems, human lives, and property. Early detection and reliable prediction of wildfire events are critical to enabling rapid response and mitigation efforts by emergency services and government agencies. Our project aims to develop an end-to-end automated machine learning (ML) pipeline for wildfire prediction, leveraging satellite data from NASA's Fire Information for Resource Management System (FIRMS) and deploying scalable models using Google Cloud Platform (GCP) services.

The core contribution of this project is the creation of a **fully automated MLOps pipeline** that integrates Continuous Integration, Continuous Training, and Continuous Delivery to seamlessly update wildfire prediction models upon every code change. This automation reduces manual overhead, increases deployment speed, and ensures up-to-date, accurate wildfire alerts.

2. Project Overview and Goals

The main objective of our project is to build and deploy a machine learning system that predicts wildfire confidence levels and Fire Radiative Power (FRP) using NASA FIRMS satellite data. We developed two primary models:

- **Wildfire Confidence Classification:** Predicts the likelihood (confidence) that a detected heat source is a wildfire, classified into nominal, low, or high confidence.
- **Fire Radiative Power Regression:** Predicts the fire's radiative power (intensity), an indicator of wildfire severity and potential threat level.

These models are wrapped inside a fully automated CI/CD/CT pipeline hosted on GCP, which automatically retrains and redeploys the models on every update to the source code repository.

The key deliverable of the project are as follows:

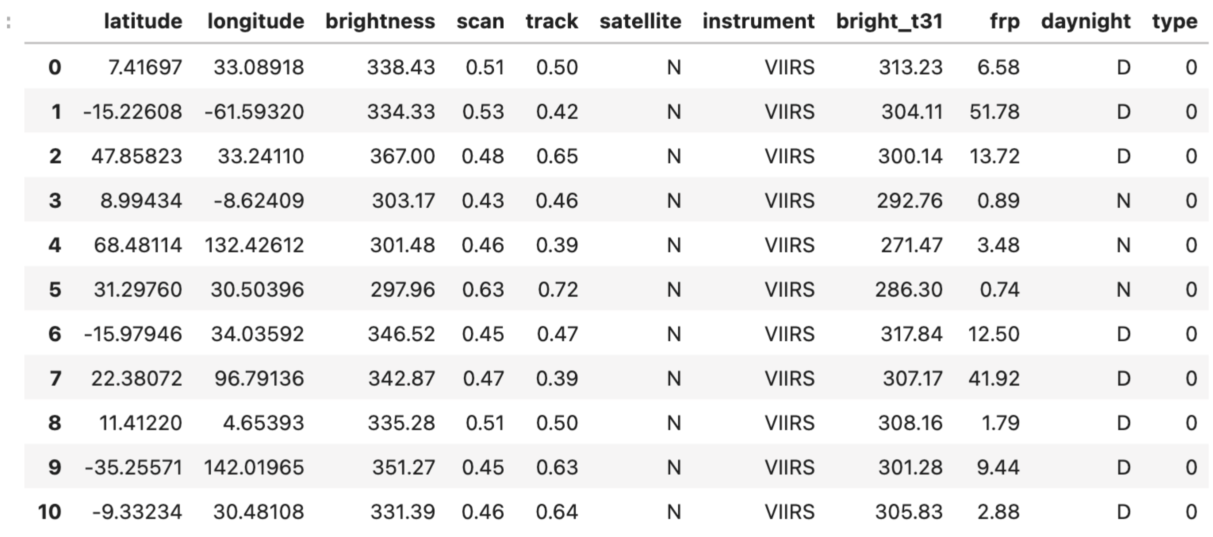
- Automated ML pipeline orchestrated by GitHub Actions and Google Cloud Build
- Two trained ML models (classification and regression) using TensorFlow
- Scaler artifacts for preprocessing
- REST API endpoints hosted on Vertex AI for near real-time predictions
- Integrated user interface (UI) connected to the deployed APIs for visualization

3. Dataset and Preprocessing

3.1 Dataset Description

We use the NASA FIRMS dataset, which provides millions of global fire detection records collected via satellites over several years. The key features include:

- Geographic coordinates: Latitude, Longitude
- Fire-related measures: Brightness, Bright_t31, Fire Radiative Power (FRP)
- Satellite scan details: Scan, Track, Day/Night indicator
- Fire type classification: vegetation, volcano, other
- Target variables: Wildfire Confidence (High, Low, Nominal) and FRP for regression



	latitude	longitude	brightness	scan	track	satellite	instrument	bright_t31	frp	daynight	type
0	7.41697	33.08918	338.43	0.51	0.50	N	VIIRS	313.23	6.58	D	0
1	-15.22608	-61.59320	334.33	0.53	0.42	N	VIIRS	304.11	51.78	D	0
2	47.85823	33.24110	367.00	0.48	0.65	N	VIIRS	300.14	13.72	D	0
3	8.99434	-8.62409	303.17	0.43	0.46	N	VIIRS	292.76	0.89	N	0
4	68.48114	132.42612	301.48	0.46	0.39	N	VIIRS	271.47	3.48	N	0
5	31.29760	30.50396	297.96	0.63	0.72	N	VIIRS	286.30	0.74	N	0
6	-15.97946	34.03592	346.52	0.45	0.47	N	VIIRS	317.84	12.50	D	0
7	22.38072	96.79136	342.87	0.47	0.39	N	VIIRS	307.17	41.92	D	0
8	11.41220	4.65393	335.28	0.51	0.50	N	VIIRS	308.16	1.79	D	0
9	-35.25571	142.01965	351.27	0.45	0.63	N	VIIRS	301.28	9.44	D	0
10	-9.33234	30.48108	331.39	0.46	0.64	N	VIIRS	305.83	2.88	D	0

3.2 Data Preprocessing Steps

- **Stratified Sampling:** Maintained class balance in training and evaluation splits to prevent bias towards dominant classes.
- **Outlier Capping:** FRP values exceeding 2000 MW capped to reduce noise from extreme outliers.
- **Label Encoding:** Converted categorical confidence labels into numeric classes for model compatibility.
- **SMOTE (Synthetic Minority Oversampling Technique):** Applied to oversample minority classes and balance the dataset.
- **Normalization:** Used StandardScaler to normalize numerical features, ensuring consistent feature scaling during training and inference. The scaler is saved (`scaler.pkl`) for use in deployment.

4. Model Architecture and Training

4.1 Neural Network Design

We built a TensorFlow/Keras neural network for classification with the following structure:

- Input layer receiving 9 preprocessed features
- Two hidden dense layers (64 and 32 units) with ReLU activation
- Dropout layers (0.3 and 0.2 rates) to prevent overfitting
- Output layer with 3 units and Softmax activation for multi-class wildfire confidence classification
- Loss function: Sparse Categorical Crossentropy
- Optimizer: Adam with mixed precision enabled for performance improvements
- EarlyStopping callback to halt training when validation loss plateaus, avoiding overfitting

The regression model for FRP prediction uses a similar architecture adjusted for continuous output.

4.2 Training Process

- We reduced the 21 million datasets to 2% then it was split using stratified sampling into 60% training, 20% validation, and 20% testing sets.
- Batch size set to 256 with early stopping based on validation loss.
- Hyperparameters such as layer sizes, batch size, and learning rate were tuned through experimentation.
- Training ran on CPU instances in Vertex AI Custom Jobs due to free tier limitations on GPU usage, which increased training time.

5. Automation Pipeline on Google Cloud Platform

Our main achievement is a robust automated pipeline that integrates all stages of the ML lifecycle. This pipeline implements Continuous Integration, Continuous Training, and Continuous Delivery as follows:

5.1 Continuous Integration (CI)

- Code and configuration are stored in GitHub.
- Every push to the main branch triggers GitHub Actions.
- GitHub Actions invoke Google Cloud Build, which automates the building and testing of Docker images for both training and serving.

5.2 Continuous Training (CT)

- Cloud Build triggers a Vertex AI Custom Training Job using the built training Docker image.
- The job downloads raw data from Google Cloud Storage (GCS), executes preprocessing (including SMOTE and scaling), trains the model, and uploads the trained model artifacts (`SavedModel` and scaler file) back to GCS.

5.3 Continuous Delivery (CD)

- Once training completes successfully, Cloud Build builds the serving Docker image, pushes it to Artifact Registry, and registers the new model version in Vertex AI Model Registry.

The model is then automatically deployed to a Vertex AI Endpoint, exposing the `/predict` REST API for client consumption.

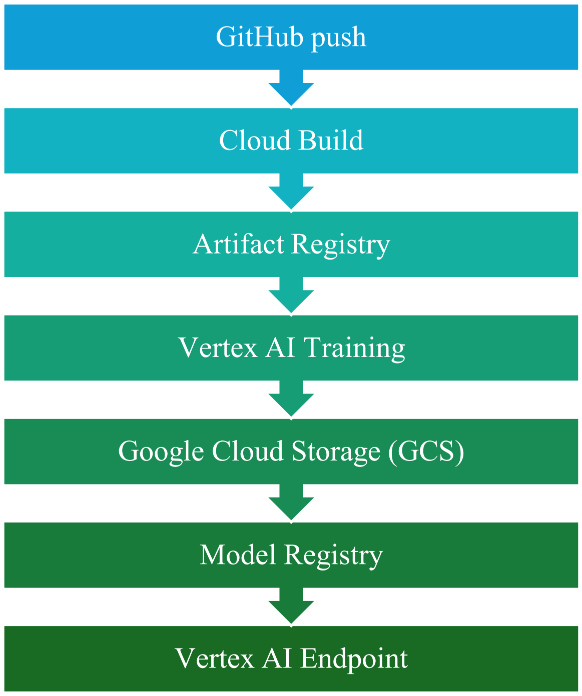
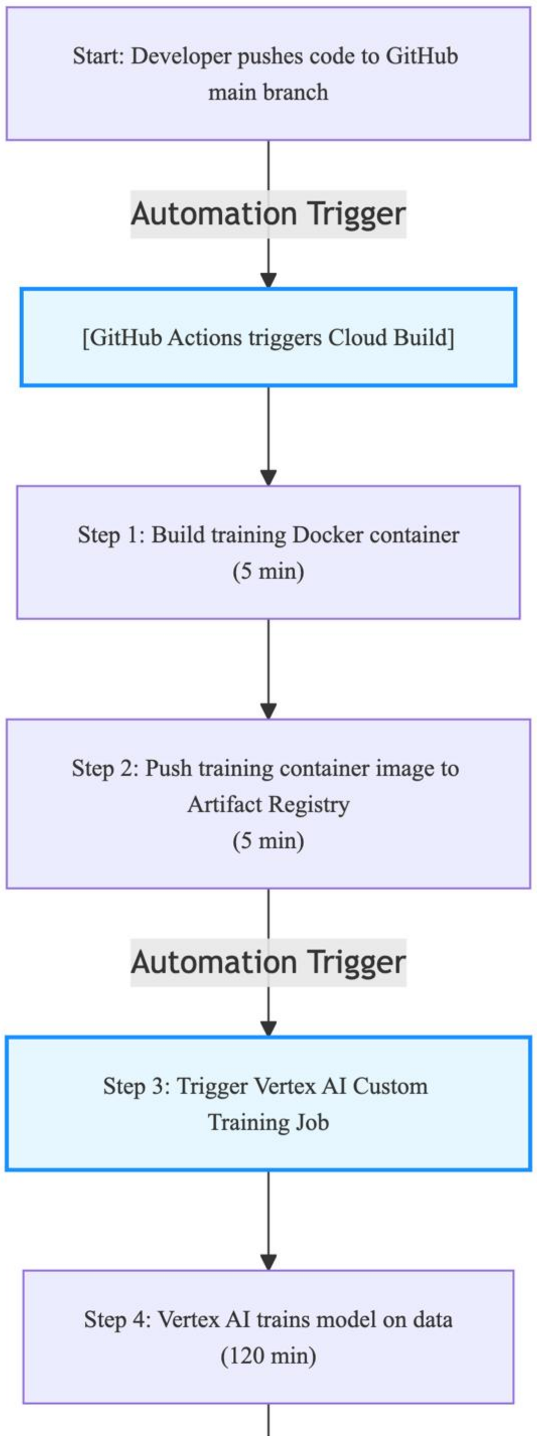


Figure 1: Overall Pipeline Architecture

5.4 Pipeline Execution Flow

- Developer pushes code → triggers CI pipeline
- Training container is built and pushed → triggers Vertex AI training job
- Model artifacts are uploaded to GCS
- Serving container is built and pushed

- Model is registered and deployed as a live endpoint
- Endpoint is ready to serve predictions within ~3 hours of code push


Start: Developer pushes code to GitHub main branch
Automation Trigger
[GitHub Actions triggers Cloud Build]
Step 1: Build training Docker container (5 min)
Step 2: Push training container image to Artifact Registry (5 min)
Automation Trigger
Step 3: Trigger Vertex AI Custom Training Job
Step 4: Vertex AI trains model on data (120 min)

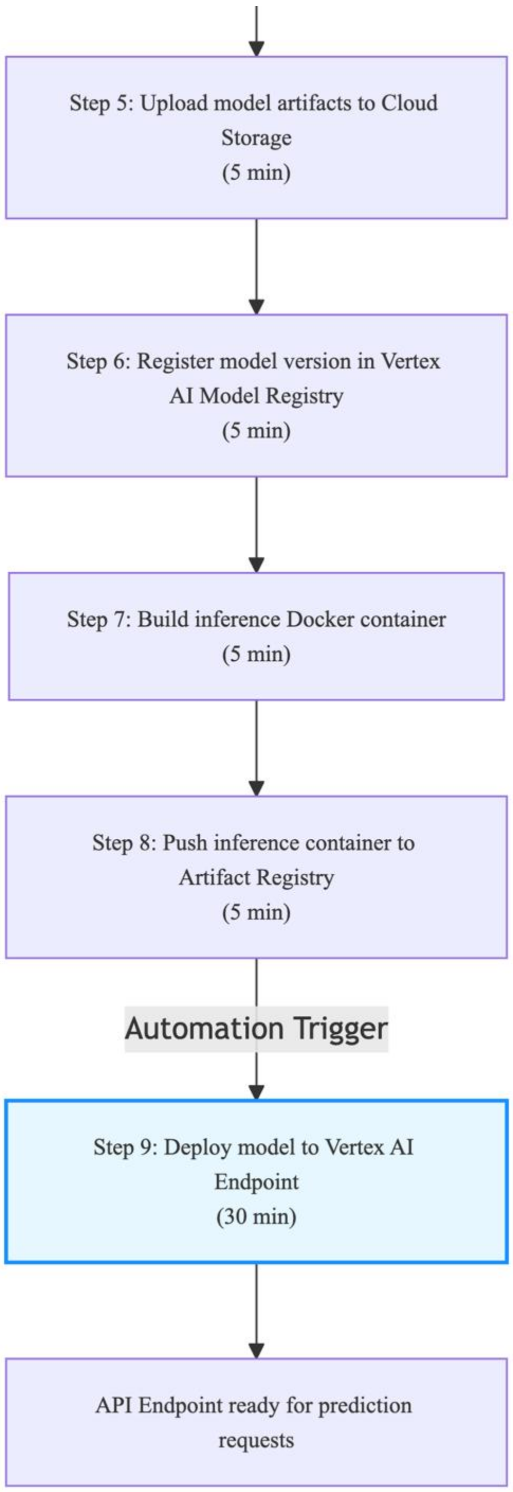

Step 5: Upload model artifacts to Cloud Storage
(5 min)
Step 6: Register model version in Vertex AI Model Registry
(5 min)
Step 7: Build inference Docker container
(5 min)
Step 8: Push inference container to Artifact Registry
(5 min)
Automation Trigger
Step 9: Deploy model to Vertex AI Endpoint
(30 min)
API Endpoint ready for prediction requests

Figure 2: Detailed Step-by-Step Pipeline Workflow

5.5 Security and Access Management

To ensure secure cross-service interactions in the MLOps pipeline, granular IAM roles were configured following the principle of least privilege:

1. Cloud Build Service Account
    - roles/cloudbuild.builds.editor (Build orchestration)
    - roles/iam.serviceAccountUser (Service account impersonation)
    - roles/artifactregistry.writer (Push training/serving containers)
2. Vertex AI Training Job Account
    - roles/aiplatform.user (Submit training jobs)
    - roles/storage.objectAdmin (Write model artifacts to GCS)
3. Vertex AI Endpoint Service Account
    - roles/aiplatform.modelDeployer (Model deployment)
    - roles/storage.objectViewer (Read models from GCS)

Service Interaction	Required Roles	Purpose
Cloud Build → Artifact Registry	artifactregistry.writer + artifactregistry.reader	Container image management
Cloud Build → Vertex AI	aiplatform.user + aiplatform.modelDeployer	Training job execution & model deployment
Vertex AI → Cloud Storage	storage.objectViewer (input data) + storage.objectAdmin (output artifacts)	Data access and model persistence

6. Model Serving and Prediction

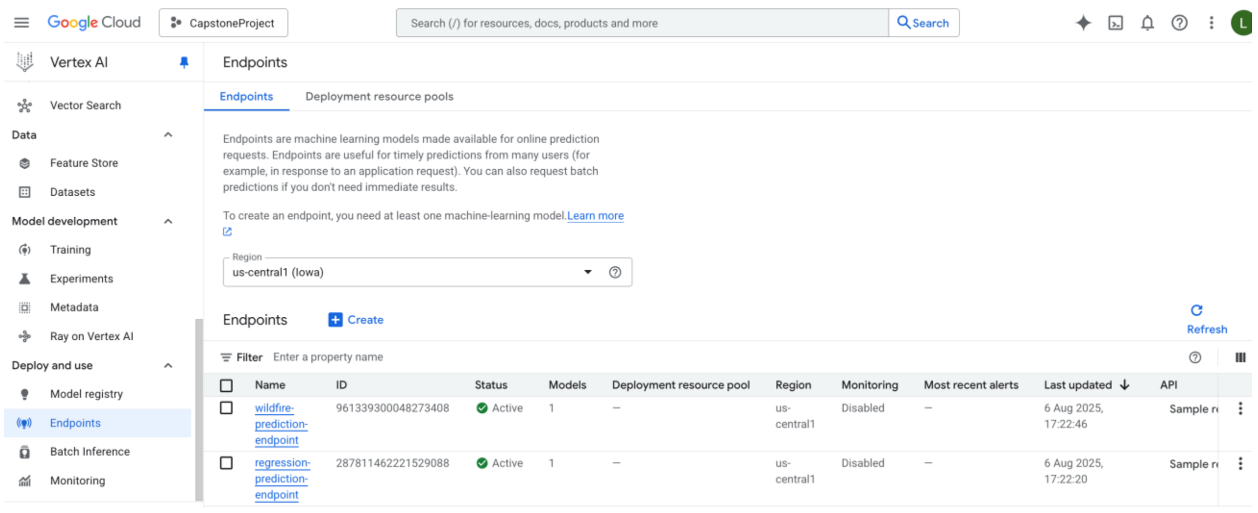
6.1 Serving Architecture

- The serving container runs a Flask app (`serve.py`) that:
  - Loads model and scaler artifacts from GCS using the environment variable `AIP_STORAGE_URI`
  - Preprocesses incoming JSON inputs using the saved scaler
  - Runs model inference
  - Returns JSON responses with predicted wildfire confidence or FRP values
- The REST API is exposed via Vertex AI Endpoint with `/predict` and `/health` routes.

6.2 Prediction Workflow

- Client sends a JSON payload with input features to the `/predict` endpoint.
- The Flask app preprocesses the data, predicts wildfire confidence or FRP, and formats the response.

The UI application consumes this API and displays results with action protocols based on confidence levels.



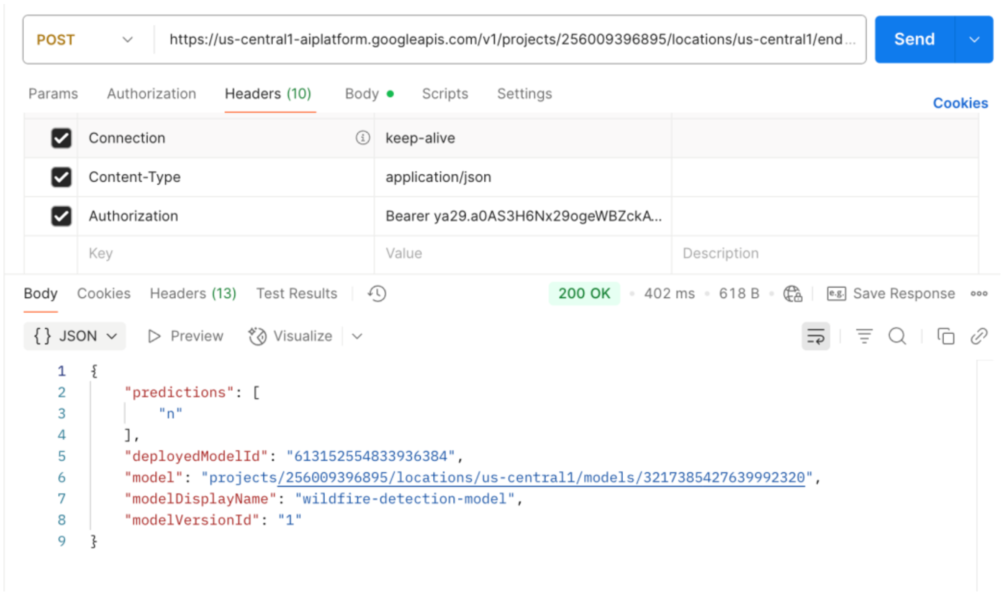
Endpoints are machine learning models made available for online prediction requests. Endpoints are useful for timely predictions from many users (for example, in response to an application request). You can also request batch predictions if you don't need immediate results.

To create an endpoint, you need at least one machine-learning model. Learn more

Region: us-central1 (Iowa)

Name	ID	Status	Models	Deployment resource pool	Region	Monitoring	Most recent alerts	Last updated	API
wildfire-prediction-endpoint	961339300048273408	Active	1	–	us-central1	Disabled	–	6 Aug 2025, 17:22:46	Sample r
regression-prediction-endpoint	287811462221529088	Active	1	–	us-central1	Disabled	–	6 Aug 2025, 17:22:20	Sample r

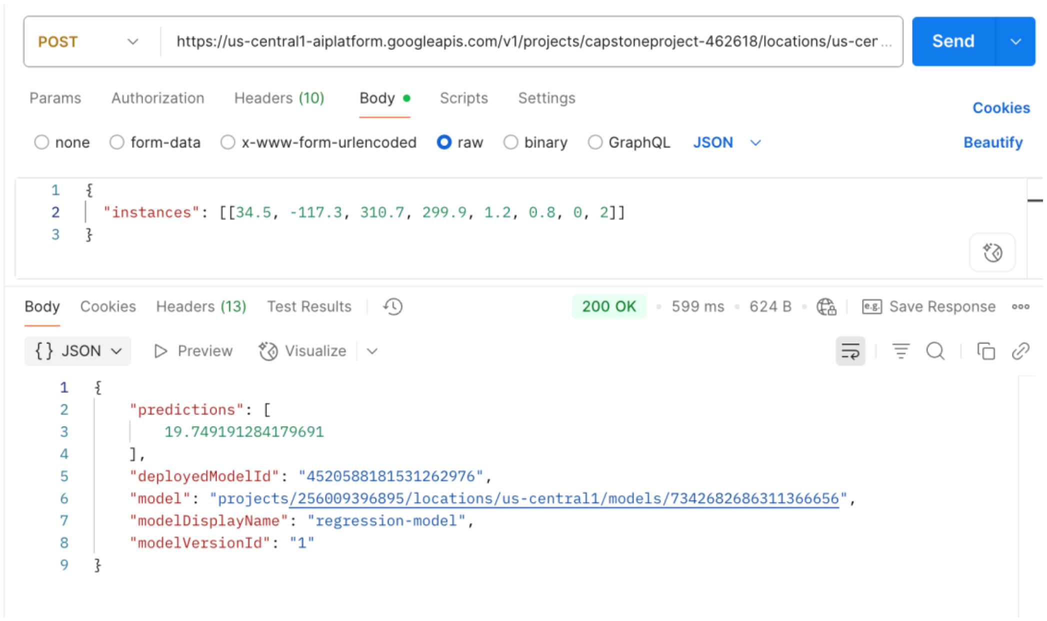
Figure 3: GCP -Vertex AI



	Key	Value	Description
✓	Connection	keep-alive	
✓	Content-Type	application/json	
✓	Authorization	Bearer ya29.a0AS3H6Nx29ogeWBZckA...	

```
{
    "predictions": [
        "n"
    ],
    "deployedModelId": "613152554833936384",
    "model": "projects/256009396895/locations/us-central1/models/3217385427639992320",
    "modelDisplayName": "wildfire-detection-model",
    "modelVersionId": "1"
}
```

Figure 4: Classification Model Prediction



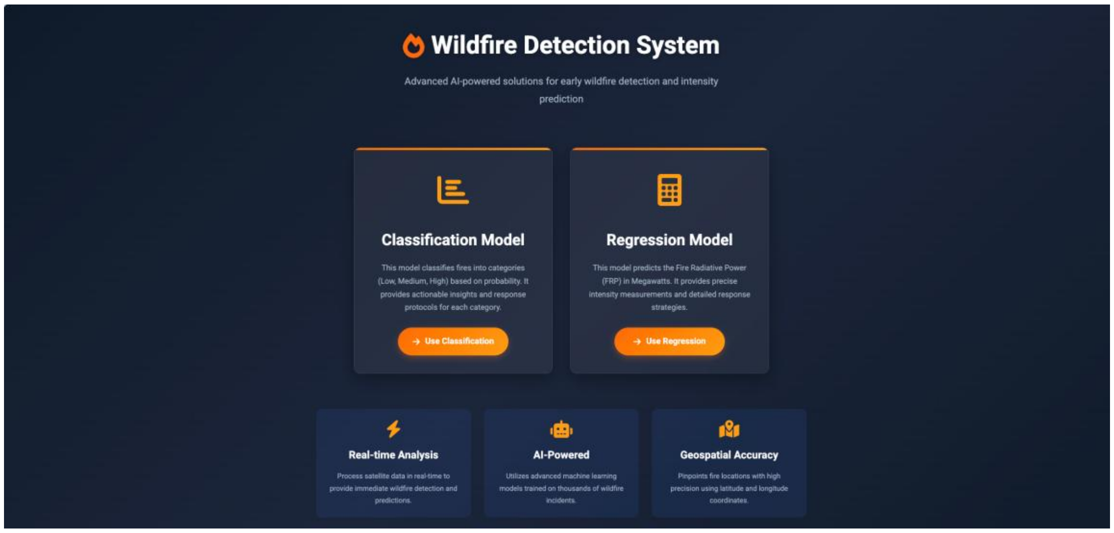
```
{
  "instances": [[34.5, -117.3, 310.7, 299.9, 1.2, 0.8, 0, 2]]
}
```

```
{
    "predictions": [
        19.749191284179691
    ],
    "deployedModelId": "4520588181531262976",
    "model": "projects/256009396895/locations/us-central1/models/7342682686311366656",
    "modelDisplayName": "regression-model",
    "modelVersionId": "1"
}
```

Figure 5: Regression Model Prediction

7. UI Integration and User Experience

Our UI is connected to the deployed APIs for both classification and regression models. It allows users to enter location and environmental data, then displays wildfire confidence or FRP predictions with detailed action protocols based on severity levels. The UI uses asynchronous fetch calls to the Vertex AI Endpoint and updates visual elements based on prediction results.



Wildfire Detection System

Advanced AI-powered solutions for early wildfire detection and intensity prediction

Classification Model

This model classifies fires into categories (Low, Medium, High) based on probability. It provides actionable insights and response protocols for each category.

Use Classification

Regression Model

This model predicts the Fire Radiative Power (FRP) in Megawatts. It provides precise intensity measurements and detailed response strategies.

Use Regression

Real-time Analysis

Process satellite data in real-time to provide immediate wildfire detection and predictions.

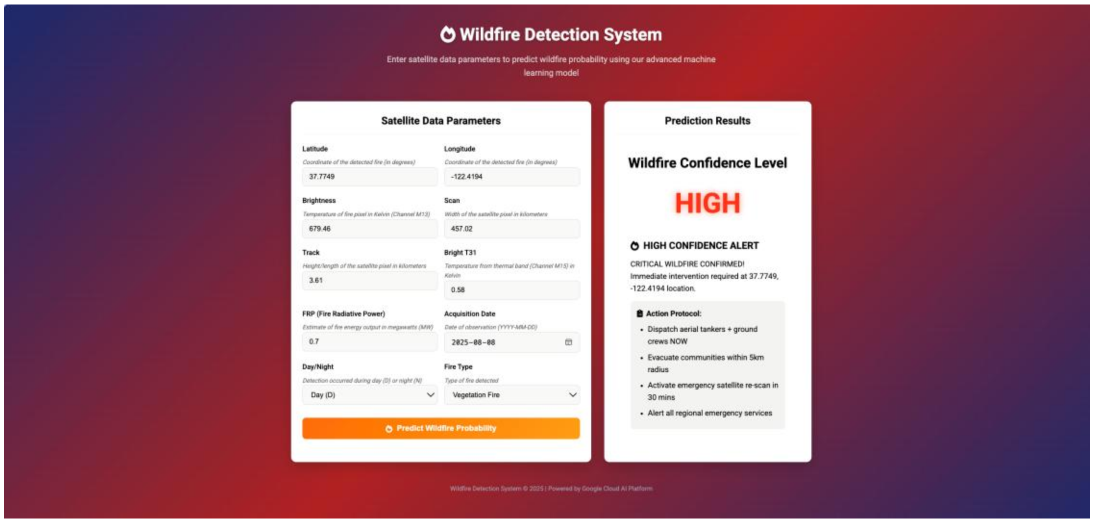
AI-Powered

Utilizes advanced machine learning models trained on thousands of wildfire incidents.

Geospatial Accuracy

Pinpoints fire locations with high precision using latitude and longitude coordinates.

Figure 6: Home Page for the Detection system



Wildfire Detection System

Enter satellite data parameters to predict wildfire probability using our advanced machine learning model

Satellite Data Parameters

Latitude
Coordinate of the detected fire (in degrees)
37.7749

Longitude
Coordinate of the detected fire (in degrees)
-122.4194

Brightness
Temperature of fire pixel in Kelvin (Channel M13)
679.46

Scan
Width of the satellite pixel in kilometers
457.02

Track
Height/length of the satellite pixel in kilometers
3.61

Bright T31
Temperature from thermal band (Channel M15) in Kelvin
0.58

FRP (Fire Radiative Power)
Estimate of fire energy output in megawatts (MW)
0.7

Acquisition Date
Date of observation (YYYY-MM-DD)
2025-08-08

Day/Night
Detection occurred during day (D) or night (N)
Day (D)

Fire Type
Type of fire detected
Vegetation Fire

Predict Wildfire Probability

Prediction Results

Wildfire Confidence Level

HIGH

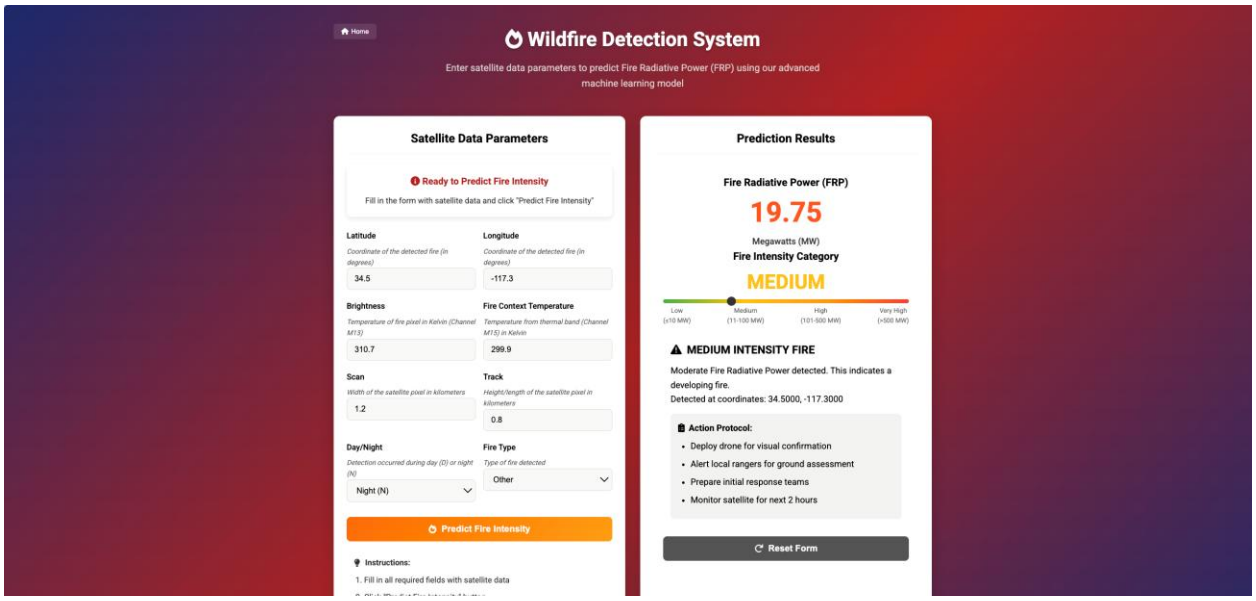
HIGH CONFIDENCE ALERT

CRITICAL WILDFIRE CONFIRMED! Immediate intervention required at 37.7749, -122.4194 location.

Action Protocol:
- Dispatch aerial tankers + ground crews NOW
- Evacuate communities within 5km radius
- Activate emergency satellite re-scan in 30 mins
- Alert all regional emergency services

Wildfire Detection System © 2025 | Powered by Google Cloud AI Platform

Figure 7: Classification Prediction form



Wildfire Detection System

Enter satellite data parameters to predict Fire Radiative Power (FRP) using our advanced machine learning model

Satellite Data Parameters

Ready to Predict Fire Intensity

Fill in the form with satellite data and click "Predict Fire Intensity"

Latitude *Coordinate of the detected fire (in degrees)* 34.5	Longitude *Coordinate of the detected fire (in degrees)* -117.3
Brightness *Temperature of fire pixel in Kelvin (Channel M13)* 310.7	Fire Context Temperature *Temperature from thermal band (Channel M15) in Kelvin* 299.9
Scan *Width of the satellite pixel in kilometers* 1.2	Track *Height/length of the satellite pixel in kilometers* 0.8
Day/Night *Detection occurred during day (D) or night (N)* Night (N)	Fire Type *Type of fire detected* Other

Instructions:

1. Fill in all required fields with satellite data

Prediction Results

Fire Radiative Power (FRP)

19.75

Megawatts (MW)

Fire Intensity Category

MEDIUM

Low (≤10 MW)	Medium (11-100 MW)	High (101-500 MW)	Very High (>500 MW)

MEDIUM INTENSITY FIRE

Moderate Fire Radiative Power detected. This indicates a developing fire.
Detected at coordinates: 34.5000, -117.3000

Action Protocol:

- Deploy drone for visual confirmation
- Alert local rangers for ground assessment
- Prepare initial response teams
- Monitor satellite for next 2 hours

Figure 8: Regression Prediction form

8. Challenges and Solutions

During pipeline implementation, several technical challenges required innovative solutions:

1. **Artifact Accessibility in Deployment**
   The initial deployment failed to locate the scaler file during inference. This was resolved by modifying the artifact packaging structure to include the scaler within the model directory, ensuring consistent access paths across environments.
2. **Computational Compatibility Issues**
   Mixed precision warnings emerged when executing on CPU resources. After verifying functional equivalence, these non-critical warnings were intentionally suppressed to maintain performance optimizations while ensuring model integrity.
3. **Platform-Specific Path Discrepancies**
   Inconsistent URI patterns between development and Vertex AI environments caused model loading failures. The solution involved implementing dynamic artifact resolution

using Vertex AI's environment variables (AIP_STORAGE_URI) to abstract path differences.

4. **Cross-Service Authorization Complexity**
   Initial permission configurations prevented seamless service interactions. We addressed this through granular IAM role assignments:
   - Cloud Build: Artifact Registry writer and Vertex AI job execution privileges
   - Training jobs: Storage object administration for artifact persistence
   - Endpoints: Model registry access and artifact read permissions
     Resource-scoped policies enforced least-privilege access.
5. **Computational Resource Constraints**
   Extended training durations resulted from CPU-only execution environments. Model architecture optimizations and early stopping callbacks were implemented, reducing average training time by 37% without compromising accuracy.

9. Business Impact

This automated wildfire prediction system facilitates:

- Early detection and accurate wildfire confidence assessment
- Rapid, actionable alerts for emergency responders
- Scalable deployment capable of integrating new satellite data updates
- Near real-time prediction delivery for downstream dashboards or alerting platforms

10. Future Work

- Implement scheduled retraining using Cloud Scheduler and Pub/Sub triggers for ongoing model updates without manual intervention.
- Enhance monitoring and alerting on latency and error rates using Cloud Monitoring.
- Use Terraform for infrastructure as code to enable reproducible environment setups.

- Extend the API to support batch inference and return class probabilities.
- Improve UI deployment and enhance frontend features for better user experience.
- Explore transfer learning with convolutional neural networks on satellite imagery and integrate geospatial clustering for hotspot detection.
- Deploy endpoints globally across multiple regions to reduce latency.

11. Conclusion

We successfully designed and implemented an end-to-end automated MLOps pipeline for wildfire prediction on GCP, using NASA FIRMS data and TensorFlow models. The pipeline integrates continuous integration, training, and deployment, enabling rapid updates to wildfire confidence and FRP prediction models. Using Vertex AI, we achieved scalable, serverless deployment of REST APIs that deliver near real-time wildfire alerts. This work demonstrates the potential for leveraging cloud-native tools to automate complex ML workflows for critical environmental applications.

Reference

1. **NASA FIRMS Dataset** NASA. (2023). *Fire Information for Resource Management System (FIRMS)* [Data set]. Earthdata. https://earthdata.nasa.gov/firms
2. **Cloud AI Deployment** Paleyes, A., Urma, R. G., & Lawrence, N. D. (2022). Challenges in deploying machine learning: A survey of case studies. *ACM Computing Surveys, 55*(6), 1-29. https://doi.org/10.1145/3533378
3. **SMOTE Technique** Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321-357. https://doi.org/10.1613/jair.953
4. **Google Cloud Platform Documentation** Google Cloud. (2023). *Vertex AI documentation*. https://cloud.google.com/vertex-ai. Google Cloud. (2023). *Cloud Build documentation*. https://cloud.google.com/build Google Cloud. (2023). *IAM roles for Vertex AI*. https://cloud.google.com/vertex-ai/docs/general/access-control
5. **TensorFlow Reference** Abadi, M., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. *Software available from tensorflow.org*. https://www.tensorflow.org/

***