Water-Intensive Crop Disease Detection and Classification Using Machine Learning

Nassim Ali-Chaouche, Jason Widjaja Djuandy,

Nail Enikeev, Vrushali Pravin Menthe

Department of Applied Data Science,

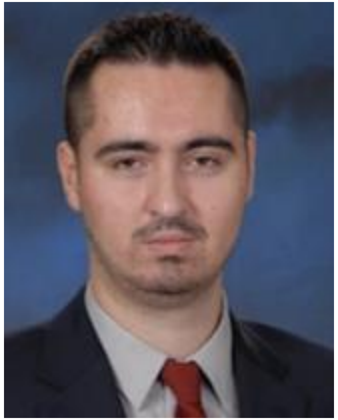
San Jose State University

DATA 298B: MSDA Project II

Dr. Jerry Gao

Team 1

Jason Djuandy



Nassim Ali-Chaouche

Nail Enikeev

Vrushali Menthe

Table of Contents

INTRODUCTION

Introduction: Project Background

- **Food and Agriculture Organization of the United Nations:**

  "FAO estimates that annually between 20 to 40 percent of global crop production is lost to pests. Each year, plant diseases cost the global economy around $220 billion, and invasive insects around US$70 billion" (FAO, 2019)

- **CNBC:**

  "There's a strained supply of rice as a result of the ongoing war in Ukraine, as well as weather woes in rice-producing economies like China and Pakistan" (Shan, 2023)

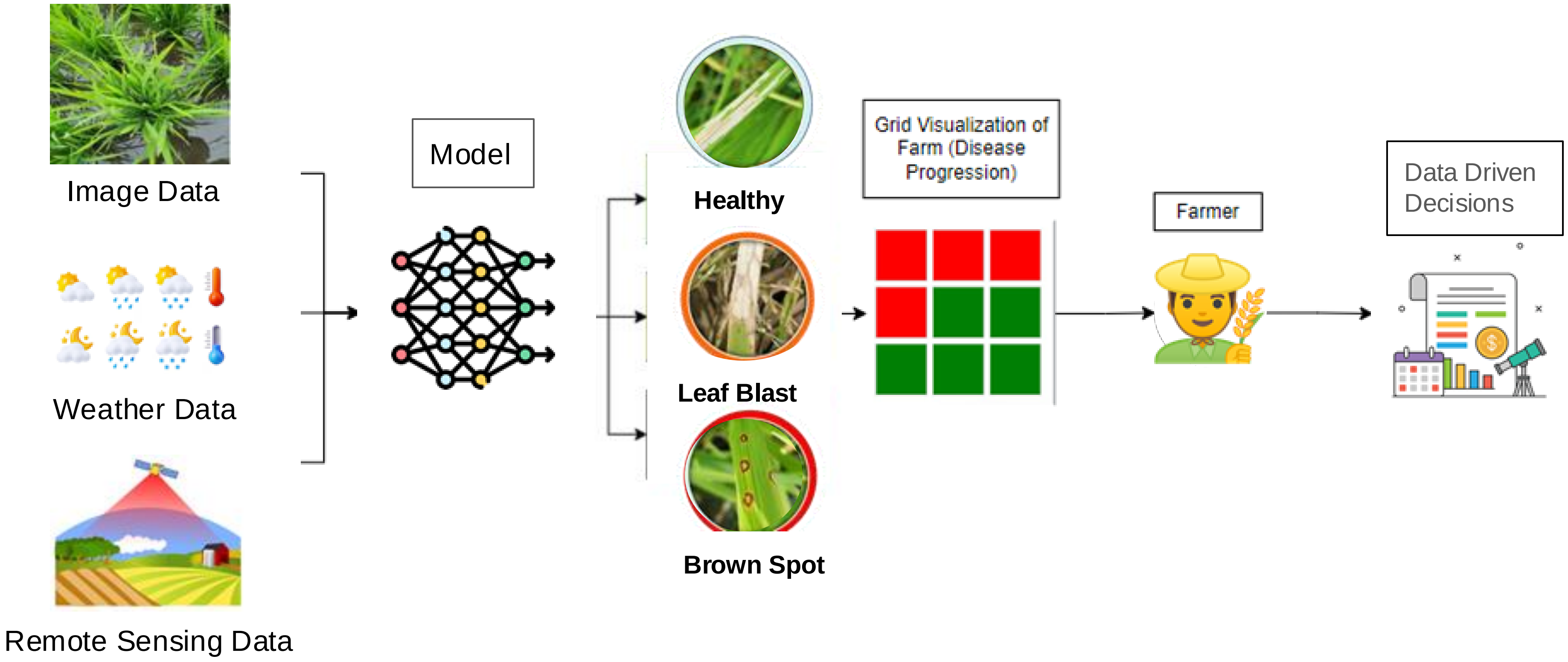
  "Rice prices surged to their highest in almost 12 years, after India's rice export ban and adverse weather conditions dented production and supplies of Asia's primary staple food, according to the UN's food agency."(Tan & Shan, 2023)

Introduction: Objective

Utilize Machine Learning Models for Water-Intensive Crop Disease Classification

- Current Research:
  - Pure Image Classification Models
  - Remote Sensing (Hyperspectral Readings) Classification Models
- Our Approach:
  - Disease Classification Model
  - Combine: Image Data + Remote Sensing Data + Weather Data
    - Use Satellite for remote sensing data
  - Compare with current approaches for effectiveness

Introduction: Problem Goal and Application



Tasks

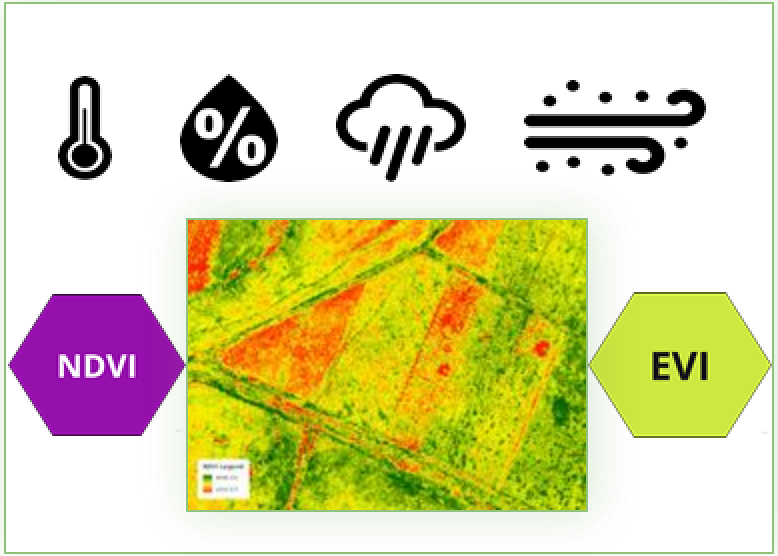
- Gather Data and Preprocess Data for Training
  - Gather images for training
  - Gather weather data and remote sensing data for each labeled image
- Create a Disease Classification Model
  - Using image classification models (Image Only)
  - Using image + weather + remote sensing data for different models (Hybrid)
  - Compare, Optimize and Fine tune selected model
- Create a Interactive Web Portal
  - Visualize disease classifications into grids.
  - Allows users such as farmers to visualize the possible disease outbreaks on their farm
  - Allows farmers to perform data driven decisions for the farm

DATA ENGINEERING

Data Engineering: Data Collection

Collected data must have combination of:

- Labeled Image Dataset
  - Rice Leaf Diseases
- Weather Data
  - Temperature
  - Humidity
  - Precipitation
  - Wind Speed
- Remote Sensing Data
  - NDVI
  - EVI



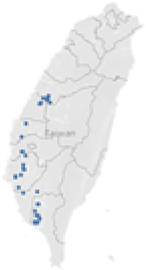
Data Engineering: Data Collection

Kaggle Dataset: Rice leaf diseases in Taiwan (927 Images)

- Contains labeled rice leaf diseases images:
  - Brown Spot
  - Leaf Blast
  - Healthy
- Majority of the Image Metadata contains:
  - Location (Geotagged)
  - Time
- Input Time and Location into API:
  - Extract Weather data using Visual Crossing API
  - Extract Remote Sensing data using Google Earth Engine API

Brown Spot

Leaf blast



Data Engineering: Data Collection

Healthy

Brown Spot

Leaf Blast

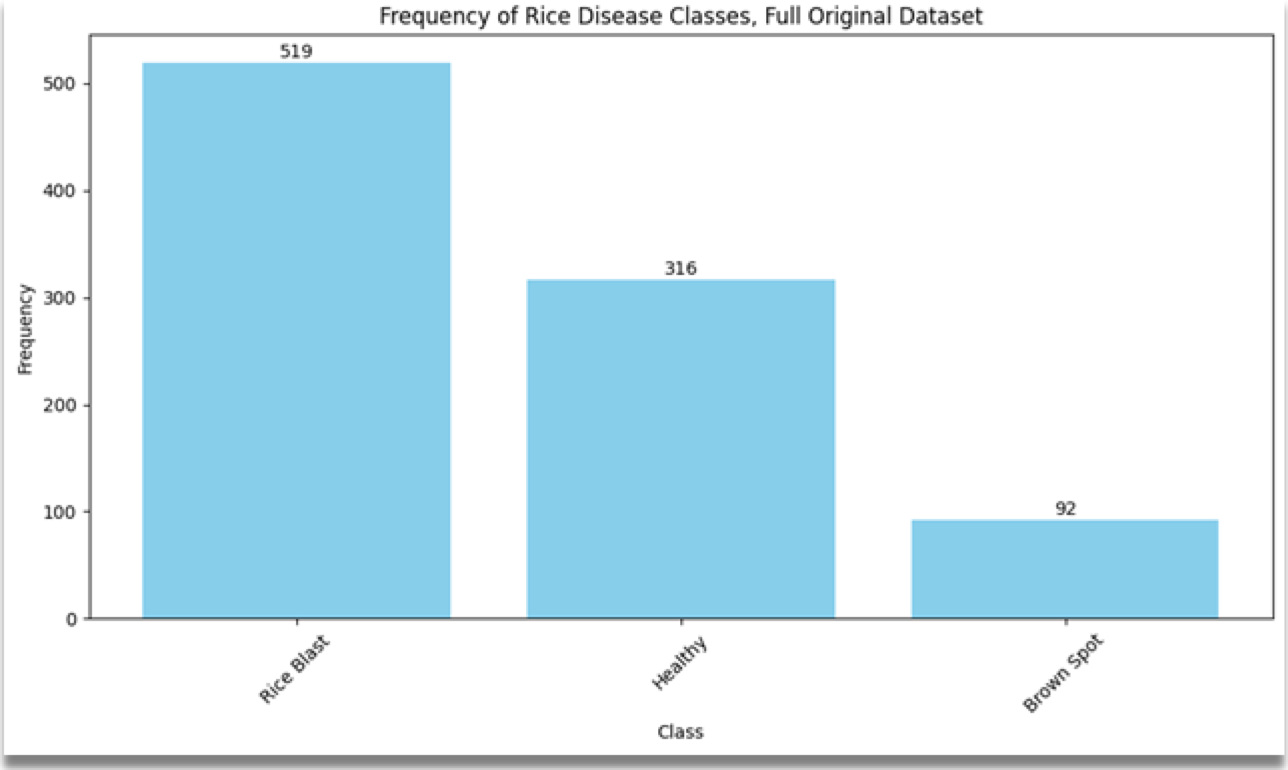
Data Engineering: Data Collection

Collected and Combined Data:

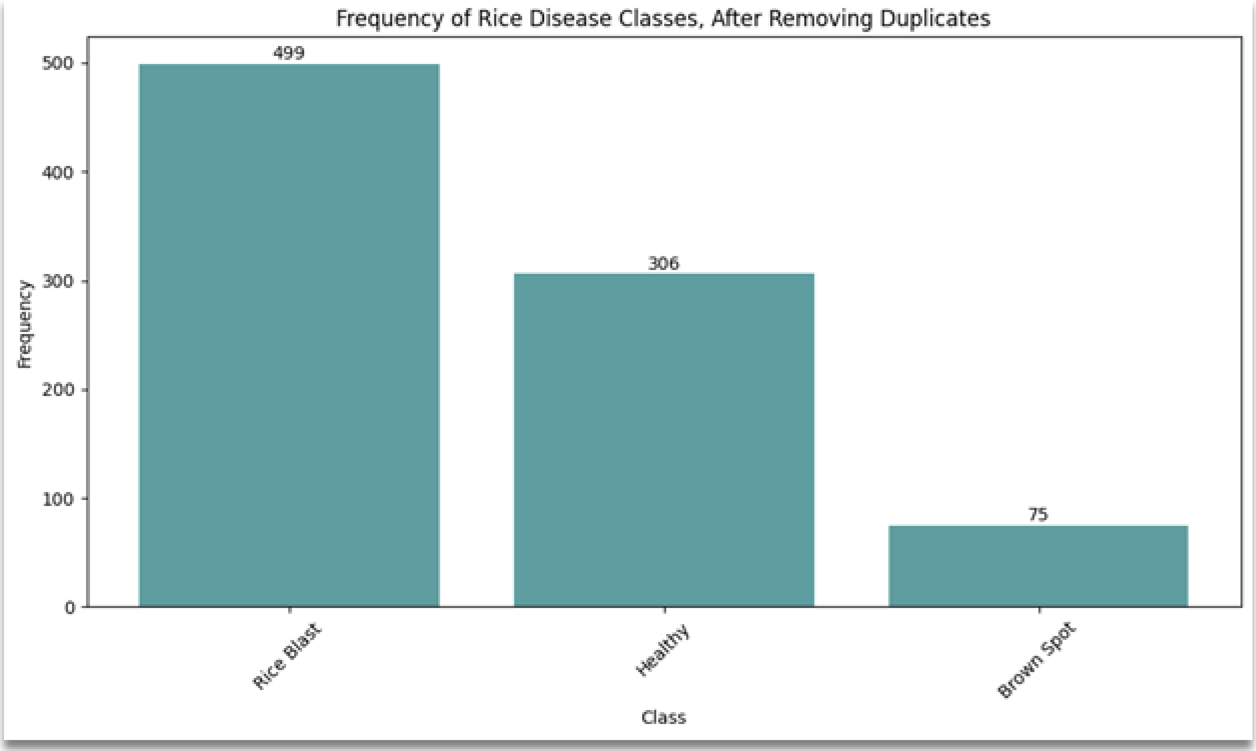
- Image
- Tabular

	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS
0	P_20181227_153331_vHDR_Auto.jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:31	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
1	P_20181227_153343_vHDR_Auto (1).jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:43	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
2	P_20181227_153711_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:11	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
3	P_20181227_153709_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:09	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
4	P_20181227_154446_vHDR_Auto (1).jpg	24.074350	120.661598	2018-12-27	Brown Spot	2018:12:27 15:44:46	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328

Data Engineering: Data Preprocessing

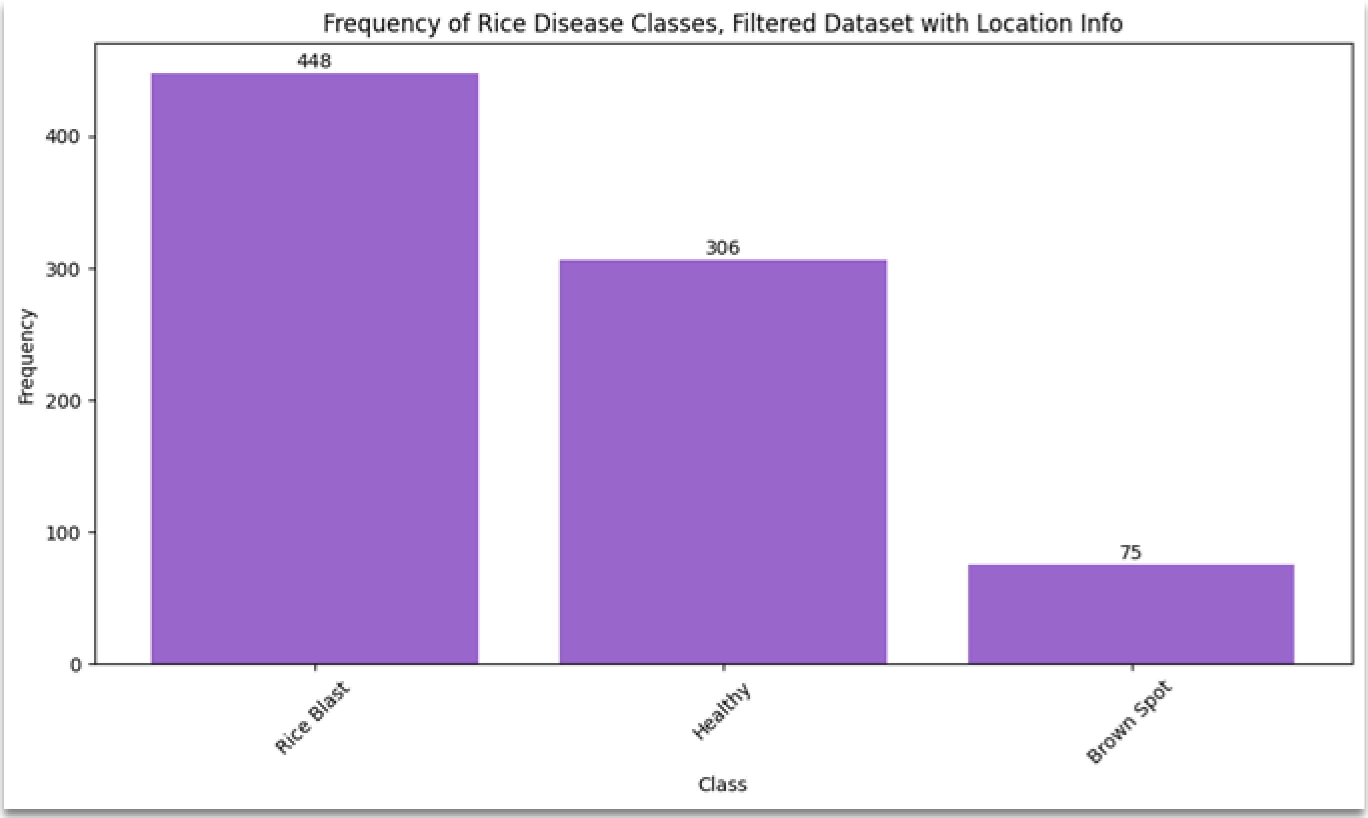


Data Engineering: Data Preprocessing



Duplicate images were detected and removed using the image’s hash

Data Engineering: Data Preprocessing



Looked into each image’s metadata and removed images without a Latitude, Longitude, and Date, which are needed later for modeling

Data Engineering: Data Preparation

- 70/15/15 train/validation/test split
  - The distribution of the class/target variable was maintained throughout the splitting of the data.
- The validation dataset is utilized during the training process to help prevent overfitting of the model.
  - An early stopping mechanism is implemented which monitors the performance of the model on the validation set
  - Prevents further training of the model when the performance on the validation set begins to degrade.

```
train_df['Class'].value_counts()

Rice Blast    313
Healthy       214
Brown Spot     53
Name: Class, dtype: int64
```

```
val_df['Class'].value_counts()

Rice Blast    67
Healthy       46
Brown Spot    11
Name: Class, dtype: int64
```

```
test_df['Class'].value_counts()

Rice Blast    68
Healthy       46
Brown Spot    11
Name: Class, dtype: int64
```

Data Engineering: Data Preparation

- To counter the imbalance of the target class, and to increase the size of the data, employed data augmentation on images in the train set only.
    - Augmenting the validation and test sets may lead to an artificially inflated performance of the model.
- Data augmentation on the train set creates diverse sets of data, which may help the model to generalize better to a wider array of unseen images.

Data Engineering: Data Preparation

```
# Distribution of the train dataset before augmentation
train_df['Class'].value_counts()
```

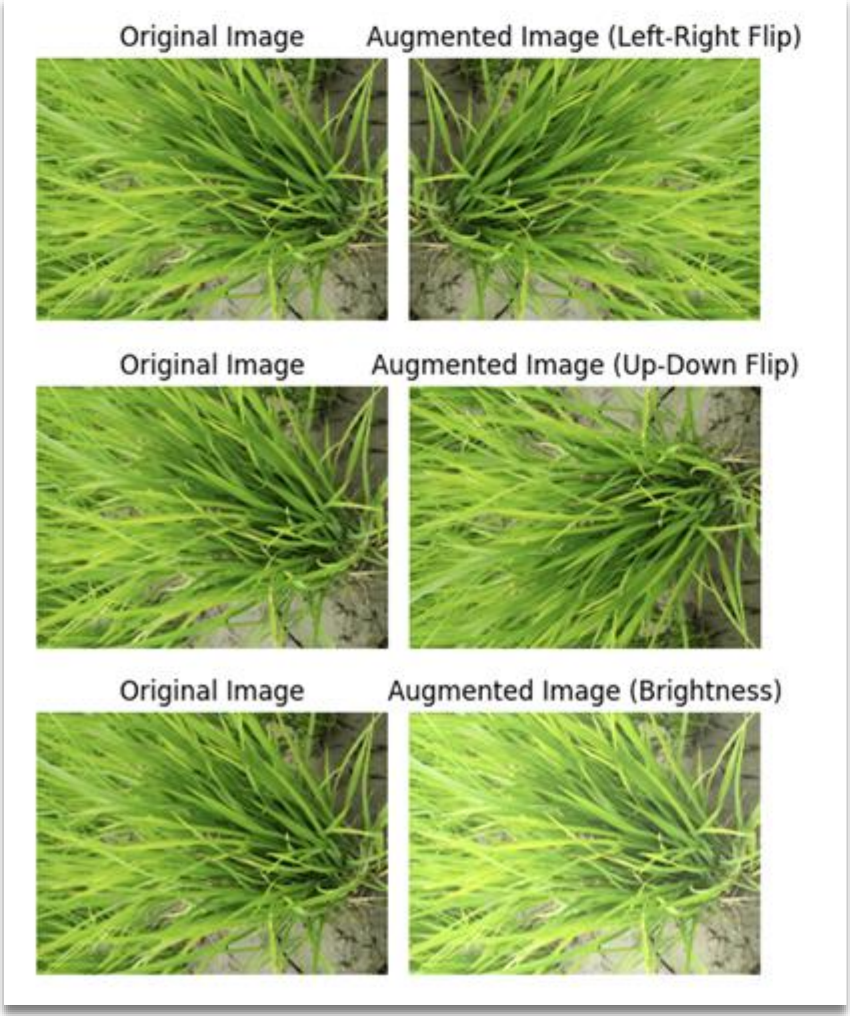
```
Rice Blast    313
Healthy       214
Brown Spot     53
Name: Class, dtype: int64
```

```
# Distribution of the train dataset after augmentation
train_df_with_augmentation_rice['Class'].value_counts()
```

```
Healthy       400
Rice Blast    400
Brown Spot    400
Name: Class, dtype: int64
```

Data Engineering: Data Preparation

- Specific augmentations: horizontal & vertical rotations, brightness changes
- Rotations: employed to make the model more robust to changes in rotation and to generalize better to a wider array of images.
- Brightness changes: images were randomly brightened or darkened with a maximum percentage change of 30.
    - Simulate various lighting conditions that may occur naturally due to environment changes, for instance due to the time of day.
    - Helps the model generalize better to images with a variety of brightness levels.


Original Image
Augmented Image (Left-Right Flip)
Original Image
Augmented Image (Up-Down Flip)
Original Image
Augmented Image (Brightness)

Data Engineering: Data Preparation - Image Data

- Some of the images were very high resolution (e.g. 4000x3000), far exceeding input sizes required for pre-trained models (e.g. 224x224 for VGG19).
  - Directly resizing these images can result in significant loss of detail and distortion of the aspect ratio.
- ‘tf.image.resize_with_pad’ - resizes the images while maintaining the original aspect ratio without distortion, preserving the fidelity of the image’s details.
- Antialiasing smooths images to prevent jaggedness that occasionally occurs when scaled down.
  - Preserving image quality is crucial when training convolutional neural network models.
- After resizing, used a ‘preprocess_input’ function of the pre-trained module in TensorFlow
  - Standardize the pixel values to a format the pre-trained model expects.
  - Ex: for VGG-19, “tensorflow.keras.applications.vgg19.preprocess_input”

Data Engineering: Data Preparation - Tabular Data

- Standardized the weather data using StandardScaler()
  - $(X - \mu) / \sigma$
  - The remote sensing data were previously scaled during initial data collection
  - All numerical data are on the same scale, and thus no bias will be introduced to the natural magnitude of the data
- Adding binary indicators to the remote sensing data that shows whether or not there was a decrease in these indices from the previous 2 periods, which might be indicative of disease

	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS	NDVI 1 Decrease	NDVI 2 Decrease	EVI 1 Decrease	EVI 2 Decrease
1	P_20181227_153343_vHDR_Auto (1).jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:43	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
2	P_20181227_153711_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:11	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
3	P_20181227_153709_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:09	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
5	P_20181227_155134_vHDR_Auto.jpg	24.074811	120.660849	2018-12-27	Brown Spot	2018:12:27 15:51:34	-0.723365	0.060275	-0.868913	1.263967	0.5403	0.3980	0.3624	0.4185	0.2271	0.2348	0	0	0	0
6	P_20181227_154452_vHDR_Auto_HP (1).jpg	24.074337	120.661612	2018-12-27	Brown Spot	2018:12:27 15:44:52	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0

Data Engineering: Data Preparation - TensorFlow Modeling

- TensorFlow's 'Dataset' API was used to transform the datasets into a format suitable for training in TensorFlow
    - 'from_tensor_slices' function with the image data, numerical data, and class labels to convert them into datasets compatible for use in TensorFlow
    - Used Dataset.zip to combine the image and numerical datasets, ensuring that future models receive two inputs
- Training, validation, and test sets were batched and prefetched to increase computational efficiency and the convergence speed of the model.

Data Engineering: Data Analytics Result

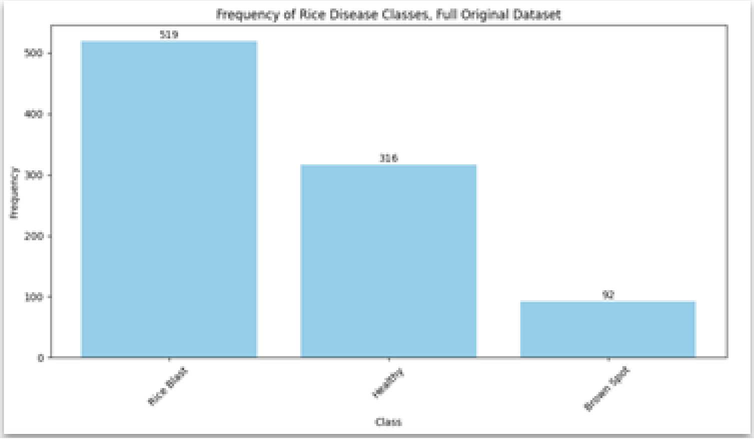
```
combined_data.head()
```

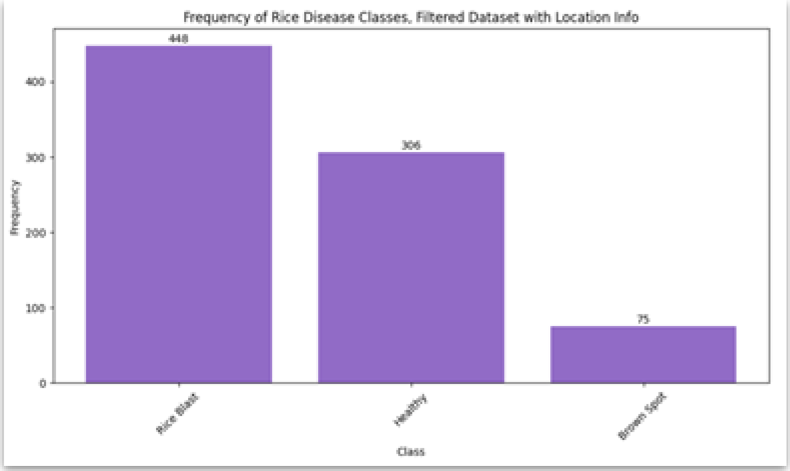
	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS
0	P_20181227_153331_vHDR_Auto.jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:31	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
1	P_20181227_153343_vHDR_Auto (1).jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:43	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
2	P_20181227_153711_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:11	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
3	P_20181227_153709_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:09	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
4	P_20181227_154446_vHDR_Auto (1).jpg	24.074350	120.661598	2018-12-27	Brown Spot	2018:12:27 15:44:46	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328

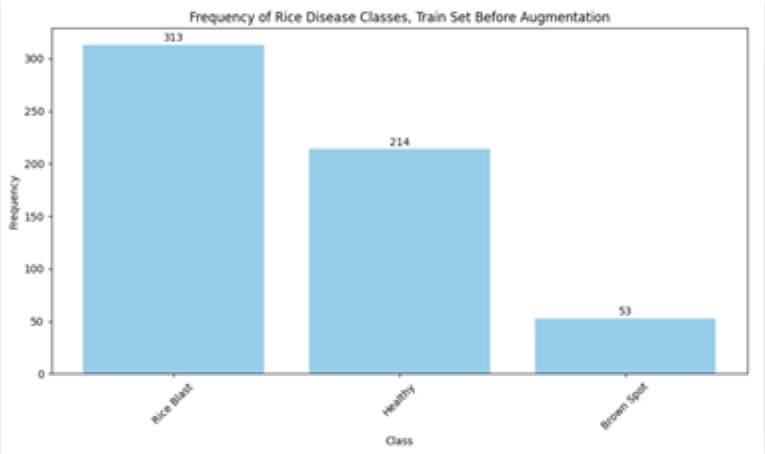
```
combined_data.describe()
```

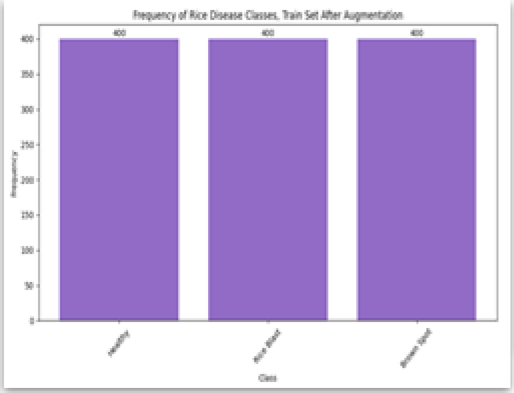
	Latitude	Longitude	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS
count	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000
mean	23.189876	120.502923	21.152550	76.597527	29.263478	20.242047	0.451565	0.405025	0.411362	0.284332	0.252975	0.251676
std	0.660420	0.128460	1.960557	2.665656	21.979152	4.591261	0.145848	0.123150	0.130431	0.108621	0.086300	0.092022
min	22.501092	120.219323	18.457143	72.257143	1.900000	13.742857	0.152000	0.170900	0.171600	0.095900	0.078400	0.097700
25%	22.515942	120.485031	19.307143	73.842857	15.231000	16.892857	0.323000	0.312000	0.275500	0.166900	0.189700	0.169400
50%	23.097482	120.487786	20.721429	76.078571	27.064000	17.728571	0.369300	0.370100	0.377900	0.268700	0.227800	0.236600
75%	24.031891	120.650942	22.764286	78.492857	28.733000	25.378571	0.591000	0.502100	0.543400	0.364500	0.301600	0.350300
max	24.119848	120.696254	27.450000	83.721429	273.800000	29.171429	0.711500	0.715400	0.603200	0.508100	0.499400	0.403400

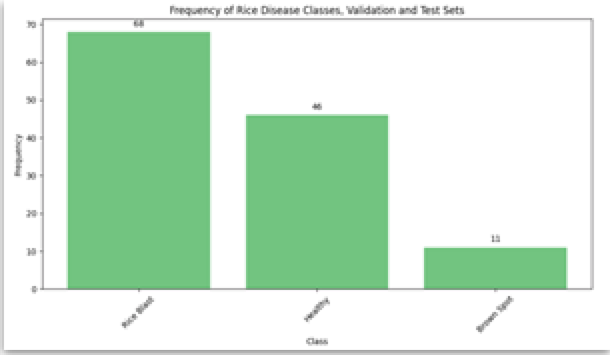
Data Engineering: Data Analytics Result









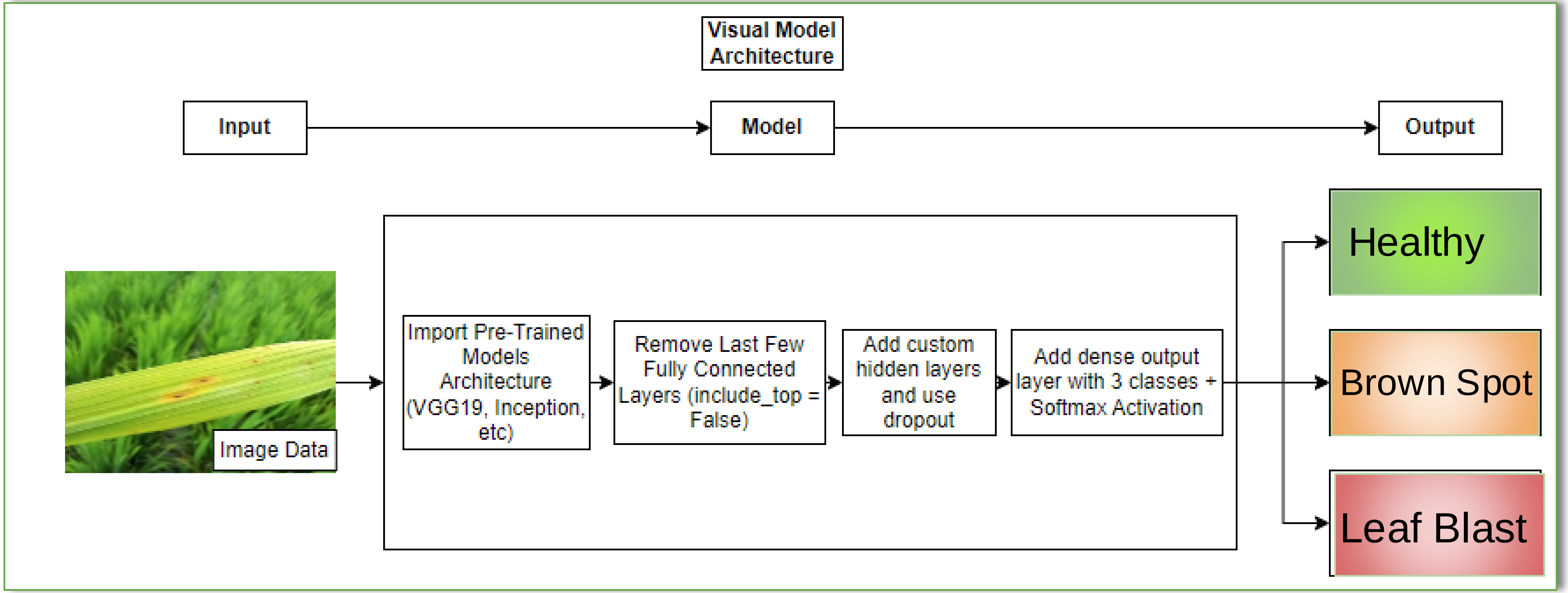


Machine Learning Models

Proposed Machine Learning Models

- VGG-19
- ResNet-50
- InceptionV3
- DenseNet121
- Hybrid Models
- Ensemble Models

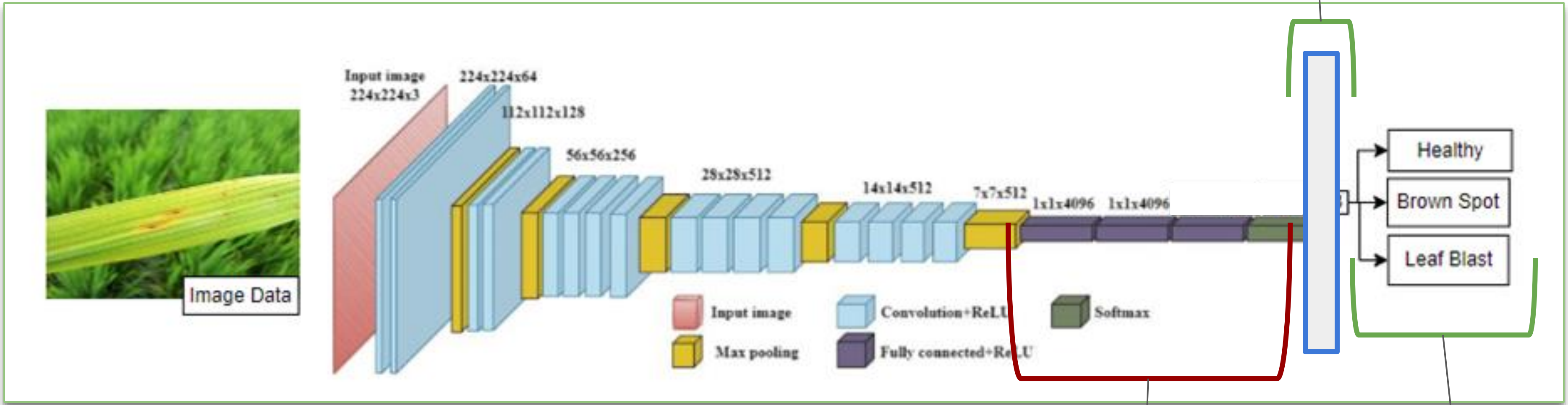
Visual-Only Model Architecture



VGG-19 Architecture

- Deep architecture, 19 layers
- Uniform 3x3 convolutions
- High computational cost

Added custom hidden and dropout layers



Removed fully connected layers

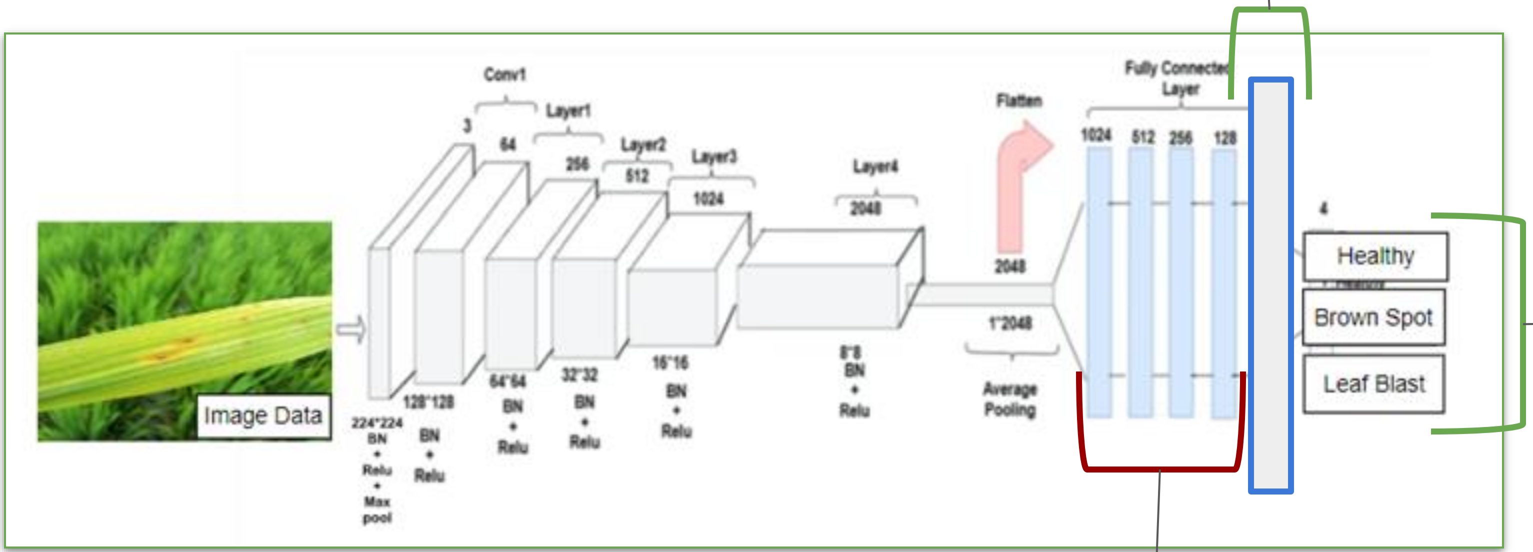
Added dense output layer with 3 classes

Note. *Image credit: Adapted from https://doi.org/10.3390/agriengineering4040056*

ResNet50 Architecture

- Residual learning, skip connections
- Solves vanishing gradient
- Efficient, deep (50 layers)

Added custom hidden and dropout layers



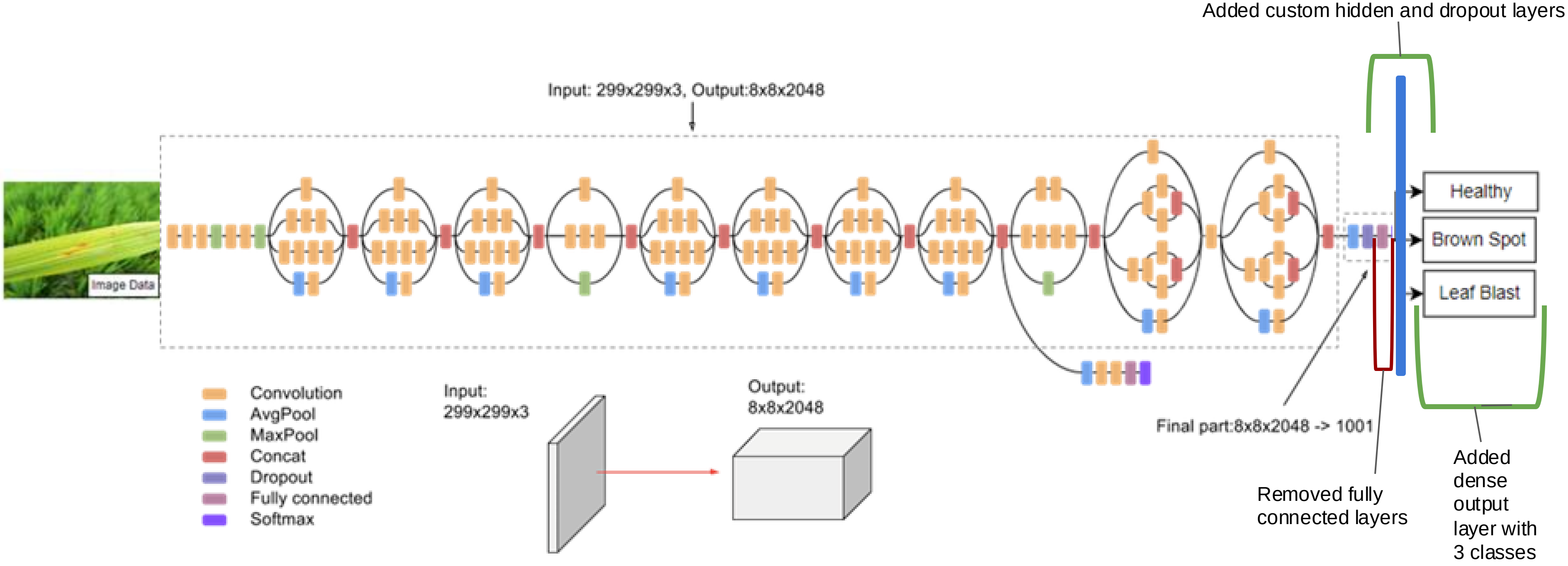
Added dense output layer with 3 classes

Removed fully connected layers

Note. *Image credit: Adapted from https://doi.org/10.1007/s00521-022-07793-2*

InceptionV3 Architecture

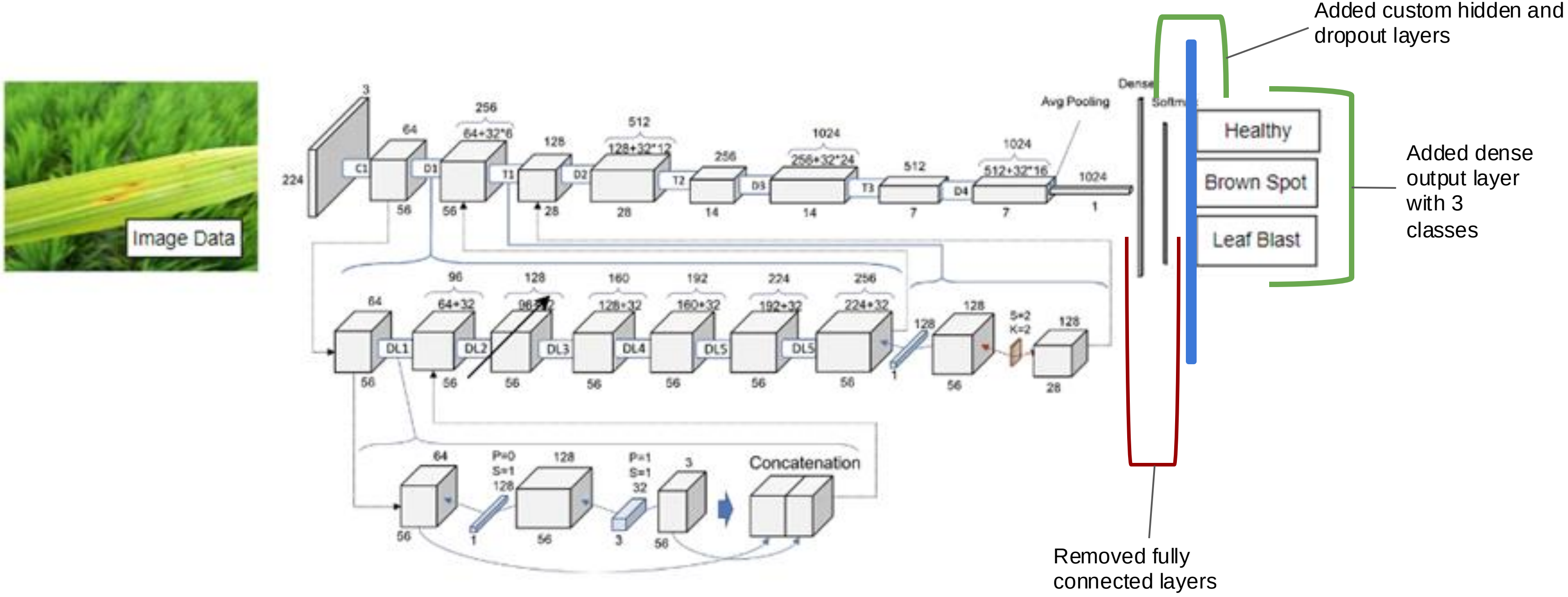
- Factorized convolutions, Inception modules
- Auxiliary classifiers, batch normalization
- Modular, scalable



Note. *Image credit: Adapted from https://doi.org/10.14569/IJACSA.2019.0100107*

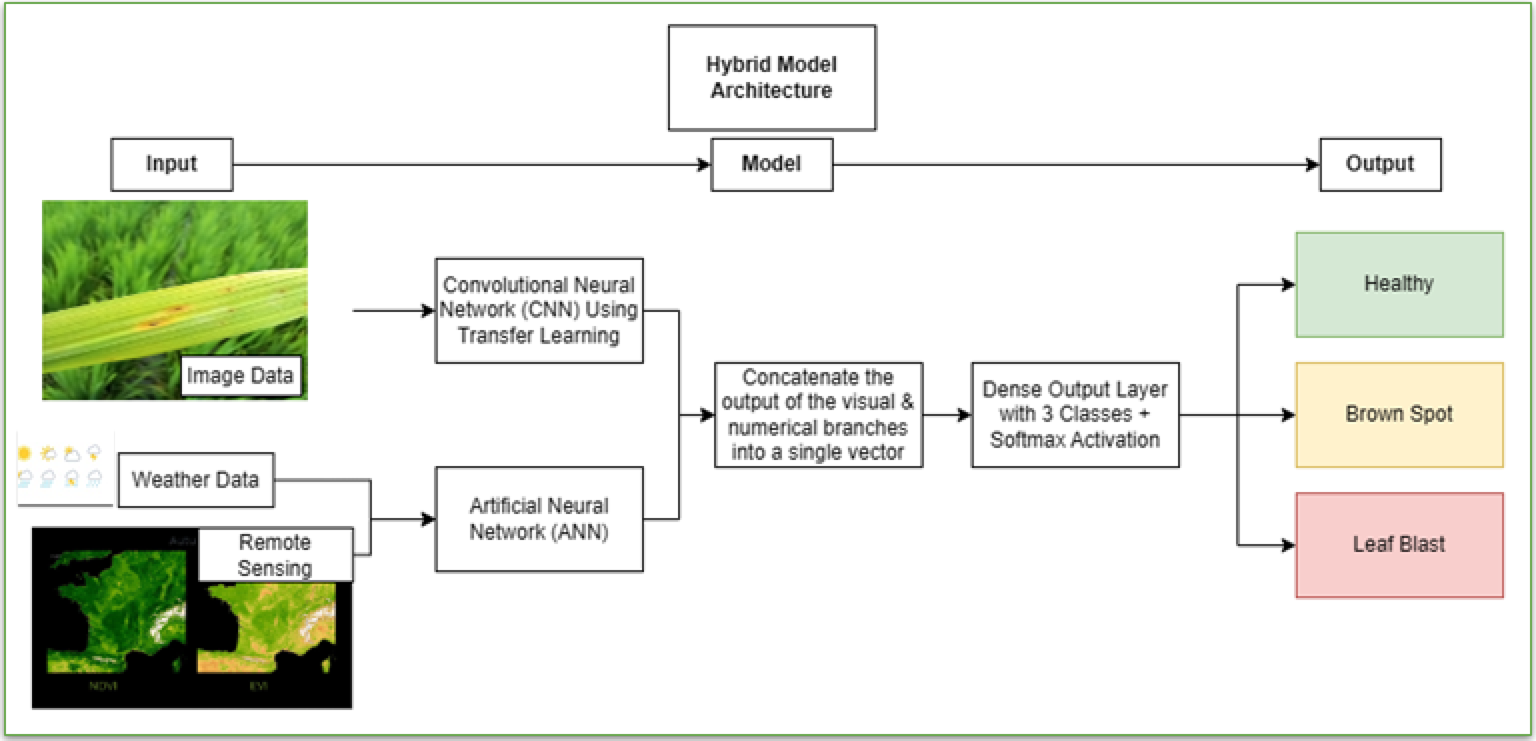
DenseNet121 Architecture

- Dense connectivity, layer-to-layer
- Parameter efficient, reduces overfitting
- Compact, effective for recognition tasks



Note. *Image credit: Adapted from https://towardsdatascience.com/understanding-and-visualizing-*

Hybrid Model Architecture



Ensemble Models

- Ensembles of both the individual hybrid models and visual-only models will be created and evaluated
  - Aggregate predictions from each model to hopefully achieve a better accuracy
- Majority Voting: each individual model makes a class prediction/vote for each instance. The class with the most votes is the final output prediction.
- Soft Voting: the average probability of each class label using each classifier is obtained, and the class with the highest probability is the final prediction

Model Evaluation Results

- Further tuning to achieve better model accuracies for both the visual-only and hybrid models
    - A greater combination of dropout rates and neurons in the hidden layers
- Modified the total number of frozen layers to further fine-tune the base transfer learning models.
    - Unfrozen layers at the end to better adapt to the features of our specific dataset
        - In the hybrid model, the visual branch includes a pre-trained base model with the last few layers unfrozen.
- The number of frozen layers involves careful consideration - more trainable layers increases computational demands and can heighten the risk of overfitting.
    - Overly adapted to the training data, compromises its ability to generalize to new instances.

Accuracies of Each Trained Model Type on the Test Set, Before and After Fine-Tuning

Base Model	**Visual-Only Model (Old/New)**	**Hybrid Model (Old/New)**
VGG-19	89.60% / 91.20%	91.20% / 96.80%
InceptionV3	92.00% / 95.20%	96.00% / 96.00%
ResNet50	93.60% / 97.60%	95.20% / **98.40%**
DenseNet121	94.40% / 96.80%	96.00% / **98.40%**
Ensemble (Soft Voting)	95.20% / 97.60%	95.20% / 96.00%
Ensemble (Majority Voting)	94.40% / 97.60%	94.40% / 96.00%

Model Evaluation Results, cont.

- More extensive experimentation with the modeling hyperparameters and fine-tuning improved accuracy for all models on the test set, except the hybrid InceptionV3.
    - Classification task - metrics are obtained by comparing the predicted & true labels
- Hybrid models outperform their visual-only counterparts, with the hybrid ResNet50 & DenseNet121 models achieving the highest accuracies.
- The models are saved in .h5 format. The **hybrid DenseNet121** model, smaller than ResNet50, is chosen as the final model.
    - Have a more lightweight model for use in the application.
    - Macro-average ROC-AUC score, OvR scheme: 0.9994, Macro-average F1-score: 0.99
    - Further underscores the exceptional predictive ability of the model across all classes.

Sample Testing Results

Raw Image + Numerical Data

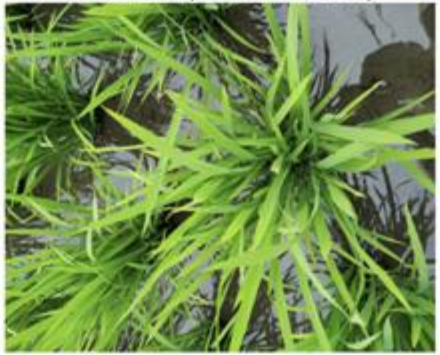


Id	Latitude	Longitude	Date and Time	Class	Date	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS
IMG_20190330_120129.jpg	22.50264222222220	120.52227713888900	2019:03:30 12:01:29	Rice Blast	2019-03-30	24.385714285714300	75.05	7.297	17.807142857142900	0.68900000000000
IMG_20190308_114943.jpg	24.098481027777800	120.66923794444400	2019:03:08 11:49:43	Healthy	2019-03-08	18.457142857142900	80.80714285714290	75.8	26.428571428571400	0.351
P_20181227_162244_vHDR_Auto_HP.jpg	24.069349222222200	120.65324747222200	2018:12:27 16:22:44	Brown Spot	2018-12-27	19.314285714285700	76.67142857142860	6.7	29.171428571428600	0.404

Model Predictions

Id	Latitude	Longitude	Date	Class Confidence Levels	Class Prediction
IMG_20190330_120129.jpg	22.50264222222220	120.52227713888900	2019-03-30	{'Rice Blast': 1.0, 'Brown Spot': 0.0, 'Healthy': 0.0}	Rice Blast
IMG_20190308_114943.jpg	24.098481027777800	120.66923794444400	2019-03-08	{'Healthy': 0.966, 'Rice Blast': 0.0339, 'Brown Spot': 0.0001}	Healthy
P_20181227_162244_vHDR_Auto_HP.jpg	24.069349222222200	120.65324747222200	2018-12-27	{'Brown Spot': 0.9988, 'Rice Blast': 0.0012, 'Healthy': 0.0}	Brown Spot

Predicted: Healthy (96.60%), True: Healthy

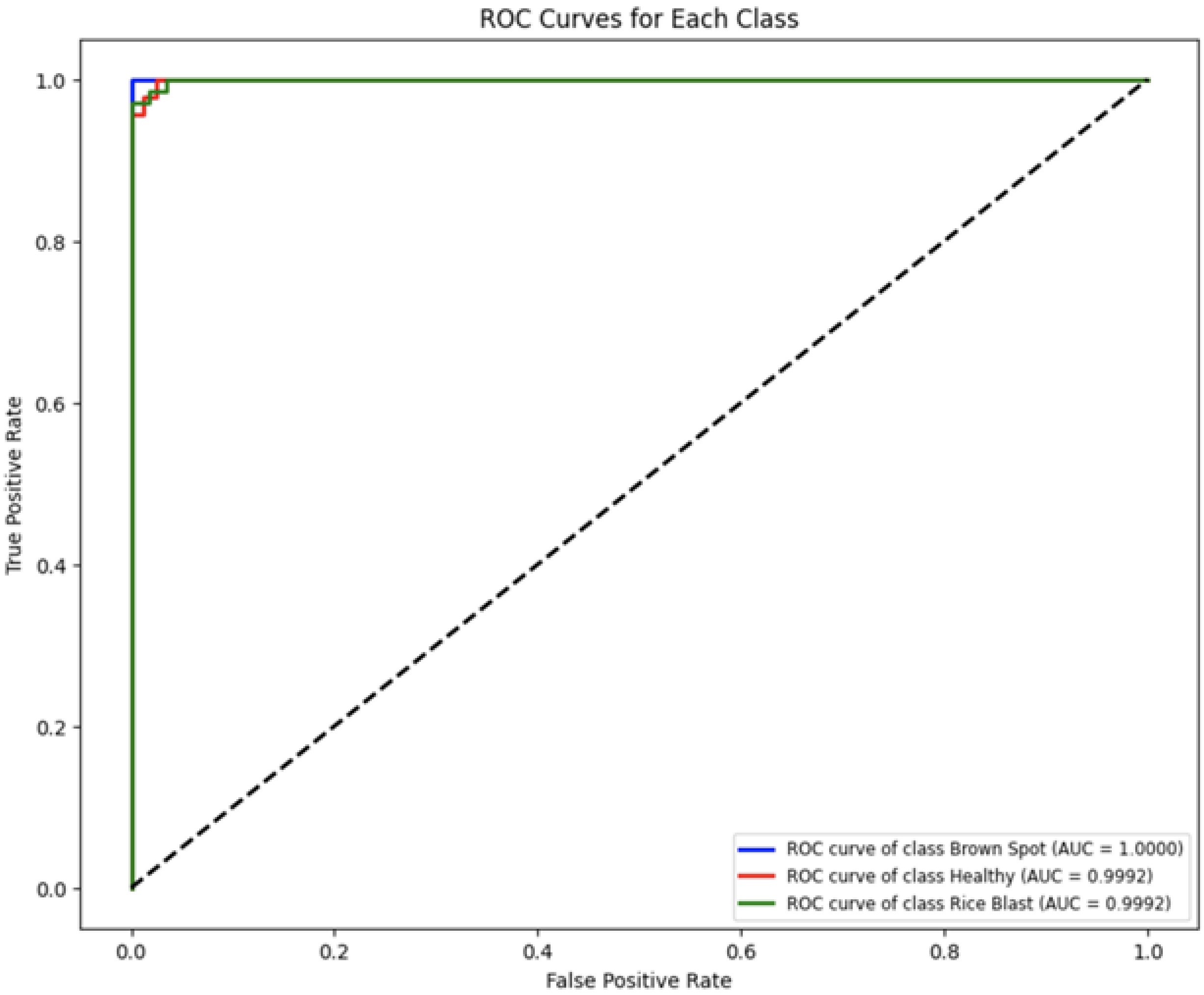


Predicted: Rice Blast (100.00%), True: Rice Blast

Predicted: Brown Spot (99.88%), True: Brown Spot

ROC Curves, OvR, Final Hybrid DenseNet121 Model

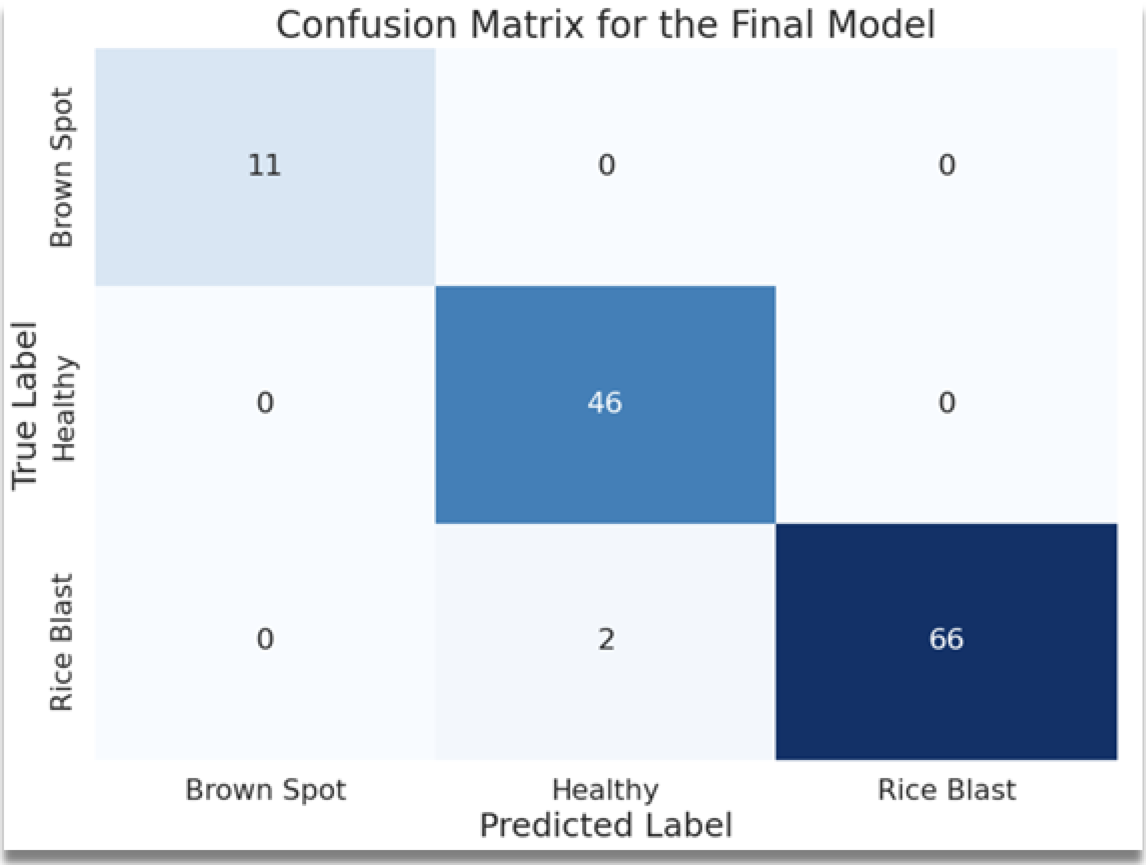
Macro- Average ROC-AUC Score: 0.9994



Classification Report, Final Hybrid DenseNet121 Model

	precision	recall	f1-score	support
Brown Spot	1.00	1.00	1.00	11
Healthy	0.96	1.00	0.98	46
Rice Blast	1.00	0.97	0.99	68
accuracy			0.98	125
macro avg	0.99	0.99	0.99	125
weighted avg	0.98	0.98	0.98	125

Confusion Matrix, Final Hybrid DenseNet121 Model



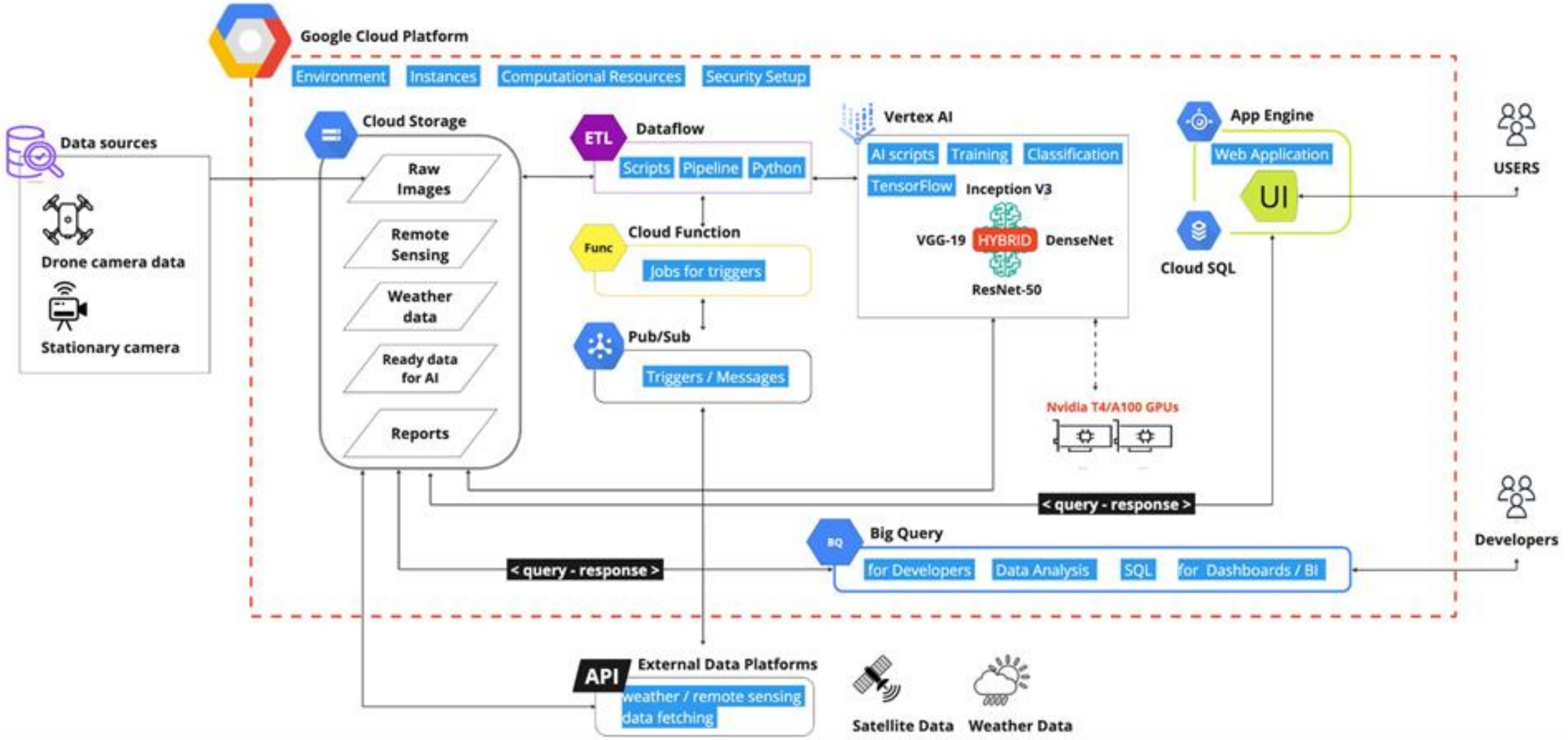
Run-Time Evaluation

Run-time performance for batch of 20 sample images with average of 3 runs:

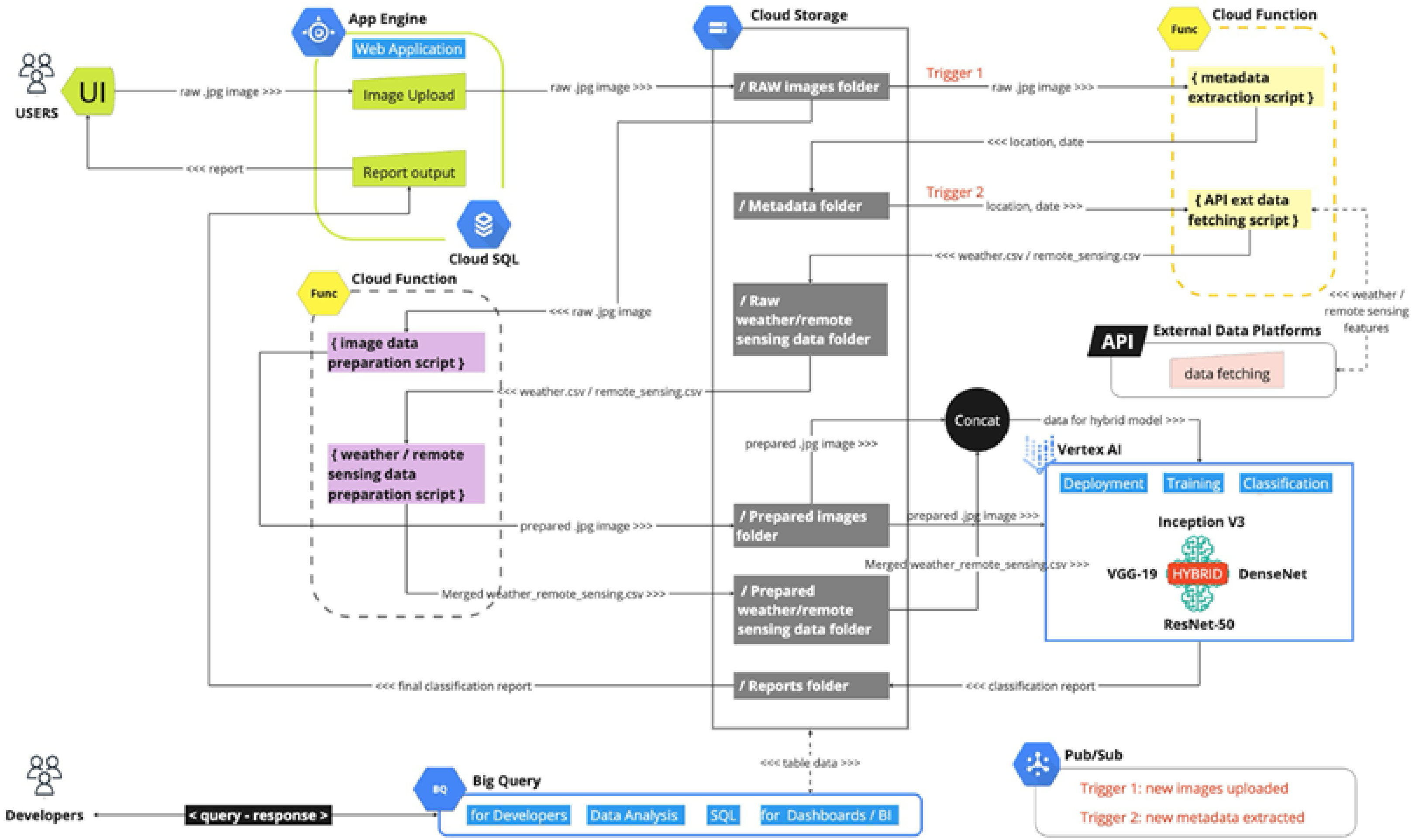
Feature	**Device Specs**	**Average Run Time (seconds)**
Extract Metadata from Images	MacBook Pro Processor: 2.3 GHz 8-Core Intel Core i9 Memory: 16GB 2667 MHz DDR4	0.1085 seconds
Obtaining Remote Sensing Data		34.4002 seconds
Obtaining Weather Data		1.4923 seconds
Combine Image Metadata, Remote Sensing and Weather Data		0.0832 seconds
Load hybrid DenseNet121 Model	Google Colab Pro V100 GPU	6.5606 seconds
Generate Predictions for all 20 instances		8.7212 seconds

System Requirements and Design

System Architecture

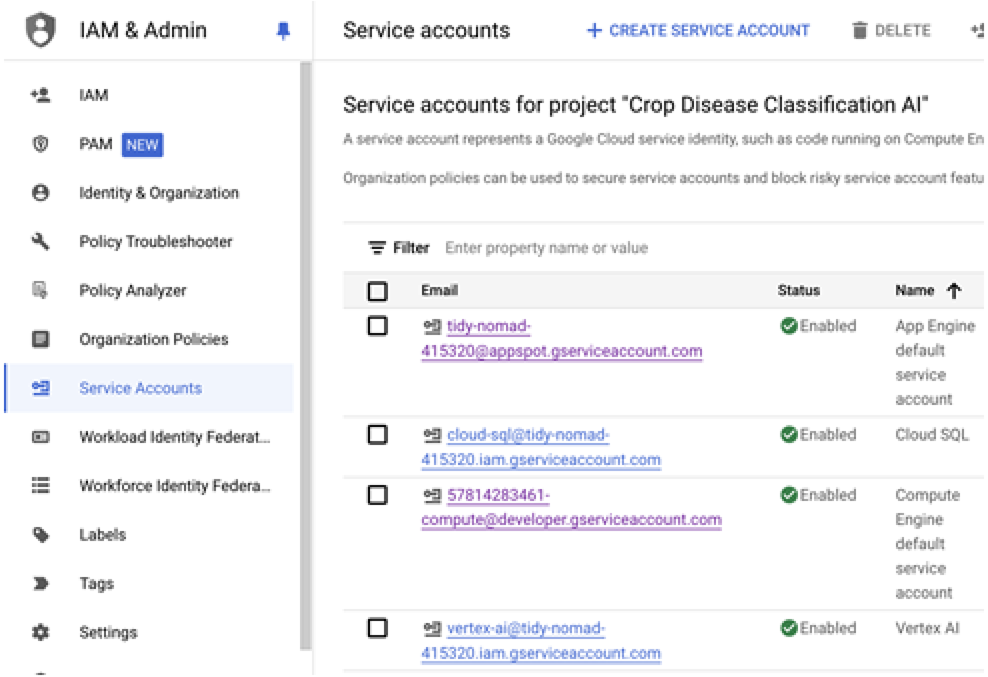


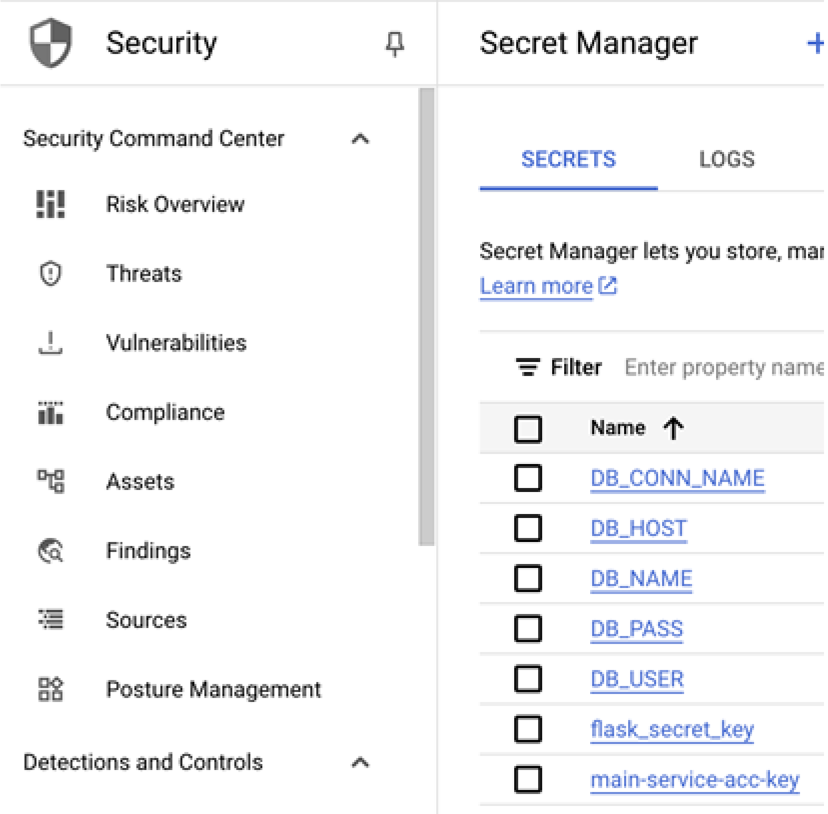
System Data Flow



Web System Design and Development

Google IAM and Secret Manager





Database tables

User Data

Image | Field | Batch | User

4 rows | Tx: Auto | DDL

WHERE | ORDER BY

	id	name	email	password	datetime
1	4	Nail	nail@cropai.com	pbkdf2:sha256:600000$Vyo1OUKgEkPpsXx7$53fcdfc819a3775540a4e91e8…	2024-03-26 01:36:42.129171
2	5	Jason	jason@crop.ai	pbkdf2:sha256:600000$yprplrGxLdEW04QH$28e56a89e9a9848181f9f208f…	2024-03-26 02:44:56.508050
3	6	Nassim	nassim@cropai.com	pbkdf2:sha256:600000$nRJFjE6uK11yfNNF$b539207b7ef1c5f36fcd9a168…	2024-03-26 02:45:11.404888
4	7	Vrushali	vrushali@cropai.com	pbkdf2:sha256:600000$eEsbIRfsGcsfvTfz$cbba9cdc643cc769a582a6611…	2024-03-26 02:45:32.394781

Field Data

Image | Field | Batch | User

4 rows | Tx: Auto | DDL

WHERE | ORDER BY

	id	user_id	name	datetime
1	84	4	002 Taiwan	2024-04-08 19:35:28.789763
2	85	4	003 Taiwan	2024-04-08 20:12:25.211833
3	86	4	004 Taiwan	2024-04-08 20:12:29.128151
4	87	4	001 Fresno	2024-04-08 20:12:34.596959

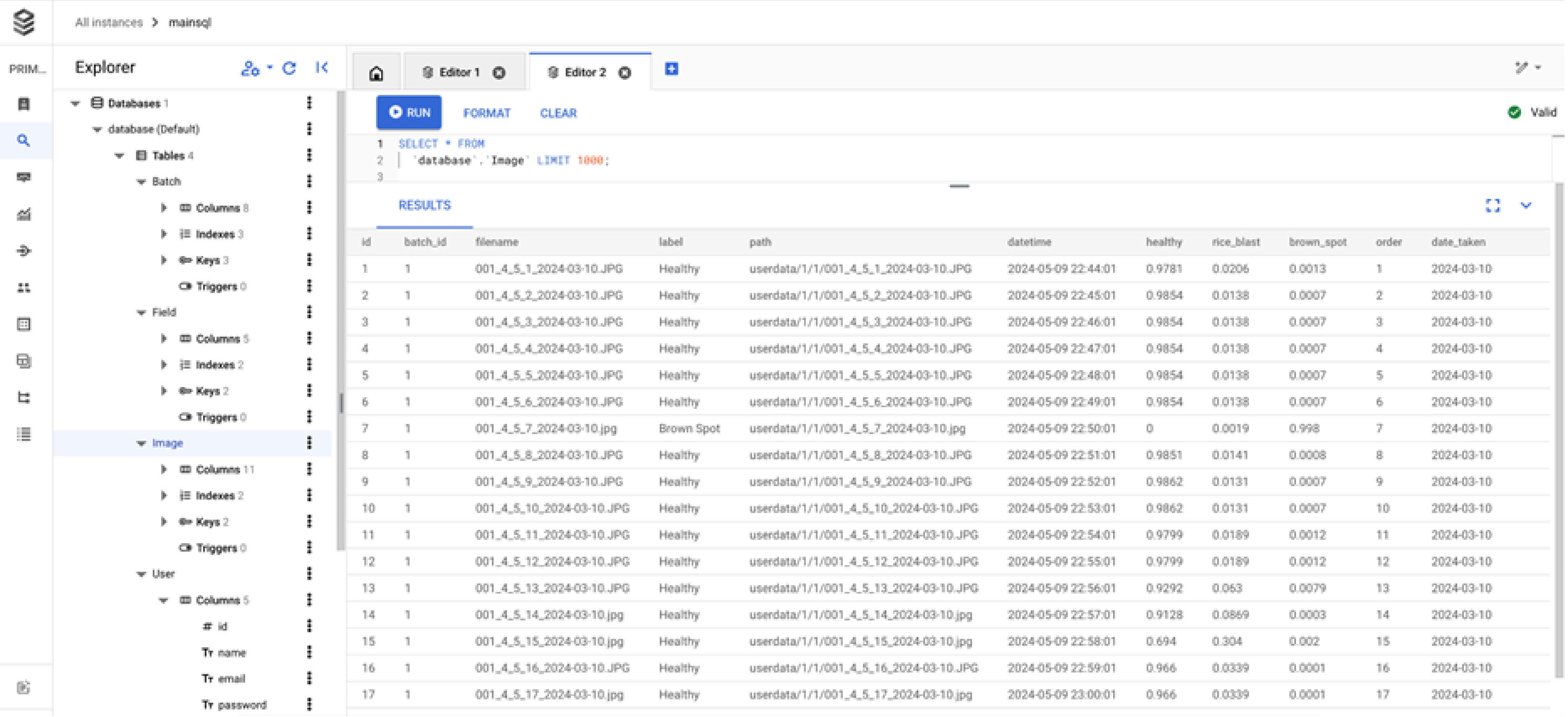
Batch Data

	id	user_id	field_id	img_qty			datetime	date_taken
1	100	4	84	20	4	5	2024-04-08 19:48:41.240163	2024-02-01
2	101	4	84	20	4	5	2024-04-08 19:56:14.679689	2024-03-01
3	102	4	84	20	4	5	2024-04-08 20:00:40.866585	2024-04-01

Image Data

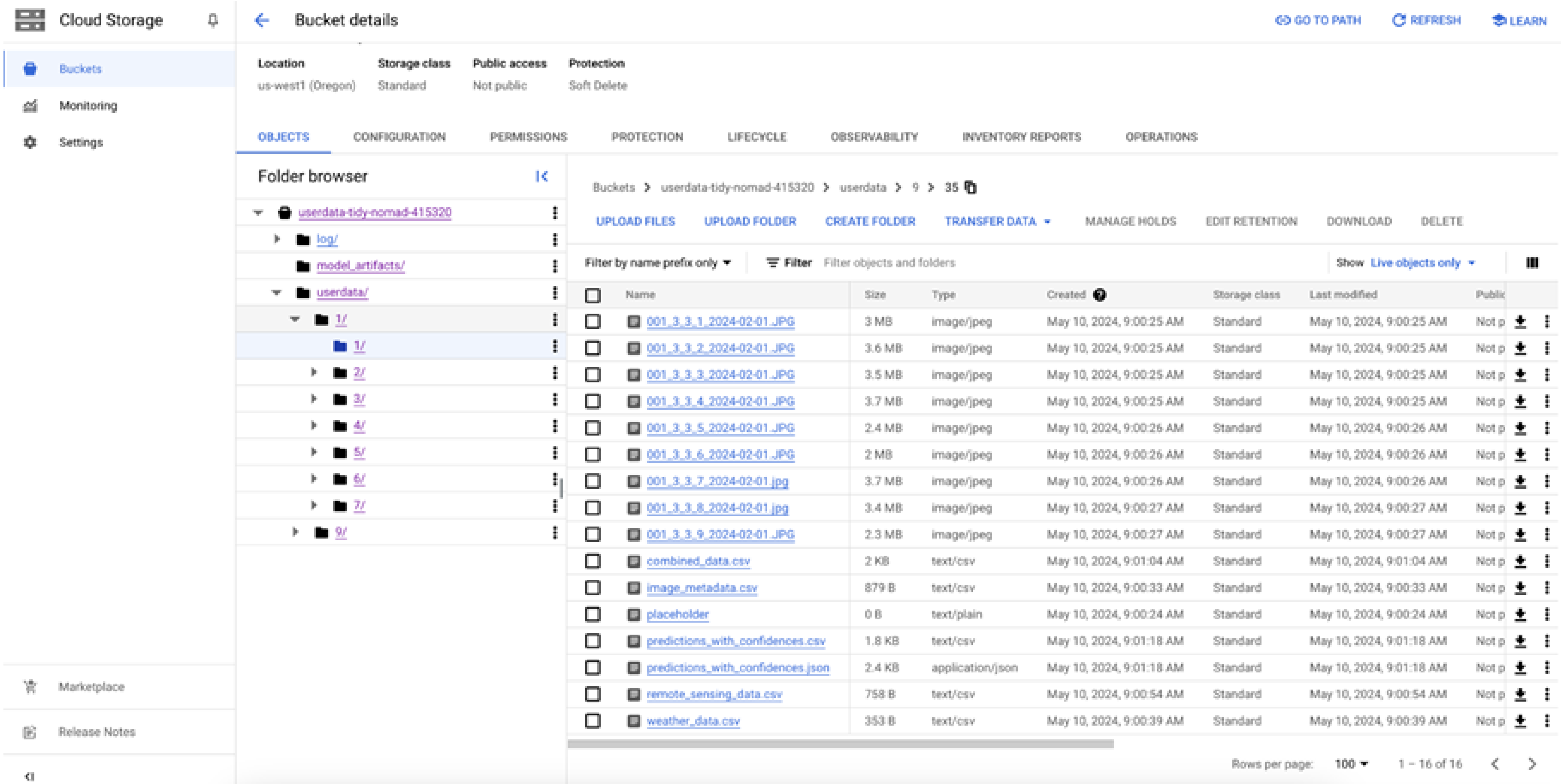
	id	batch_id	filename	label	path	datetime	healthy	rice_blast	brown_spot	order	date_taken
1	895	100	001_4_5_1_202…	Healthy	userdata/84/10…	2024-04-08 19…	0.9781	0.0206	0.0013	1	2024-02-01
2	896	100	001_4_5_2_202…	Healthy	userdata/84/10…	2024-04-08 19…	0.9854	0.0138	0.0007	2	2024-02-01
3	897	100	001_4_5_3_202…	Healthy	userdata/84/10…	2024-04-08 19…	0.9854	0.0138	0.0007	3	2024-02-01
4	898	100	001_4_5_4_202…	Healthy	userdata/84/10…	2024-04-08 19…	0.9854	0.0138	0.0007	4	2024-02-01
5	899	100	001_4_5_5_202…	Healthy	userdata/84/10…	2024-04-08 19…	0.9854	0.0138	0.0007	5	2024-02-01
6	900	100	001_4_5_6_202…	Healthy	userdata/84/10…	2024-04-08 19…	0.9854	0.0138	0.0007	6	2024-02-01

Google Cloud SQL



id	batch_id	filename	label	path	datetime	healthy	rice_blast	brown_spot	order	date_taken
1	1	001_4_5_1_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_1_2024-03-10.JPG	2024-05-09 22:44:01	0.9781	0.0206	0.0013	1	2024-03-10
2	1	001_4_5_2_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_2_2024-03-10.JPG	2024-05-09 22:45:01	0.9854	0.0138	0.0007	2	2024-03-10
3	1	001_4_5_3_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_3_2024-03-10.JPG	2024-05-09 22:46:01	0.9854	0.0138	0.0007	3	2024-03-10
4	1	001_4_5_4_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_4_2024-03-10.JPG	2024-05-09 22:47:01	0.9854	0.0138	0.0007	4	2024-03-10
5	1	001_4_5_5_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_5_2024-03-10.JPG	2024-05-09 22:48:01	0.9854	0.0138	0.0007	5	2024-03-10
6	1	001_4_5_6_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_6_2024-03-10.JPG	2024-05-09 22:49:01	0.9854	0.0138	0.0007	6	2024-03-10
7	1	001_4_5_7_2024-03-10.jpg	Brown Spot	userdata/1/1/001_4_5_7_2024-03-10.jpg	2024-05-09 22:50:01	0	0.0019	0.998	7	2024-03-10
8	1	001_4_5_8_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_8_2024-03-10.JPG	2024-05-09 22:51:01	0.9851	0.0141	0.0008	8	2024-03-10
9	1	001_4_5_9_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_9_2024-03-10.JPG	2024-05-09 22:52:01	0.9862	0.0131	0.0007	9	2024-03-10
10	1	001_4_5_10_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_10_2024-03-10.JPG	2024-05-09 22:53:01	0.9862	0.0131	0.0007	10	2024-03-10
11	1	001_4_5_11_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_11_2024-03-10.JPG	2024-05-09 22:54:01	0.9799	0.0189	0.0012	11	2024-03-10
12	1	001_4_5_12_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_12_2024-03-10.JPG	2024-05-09 22:55:01	0.9799	0.0189	0.0012	12	2024-03-10
13	1	001_4_5_13_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_13_2024-03-10.JPG	2024-05-09 22:56:01	0.9292	0.063	0.0079	13	2024-03-10
14	1	001_4_5_14_2024-03-10.jpg	Healthy	userdata/1/1/001_4_5_14_2024-03-10.jpg	2024-05-09 22:57:01	0.9128	0.0869	0.0003	14	2024-03-10
15	1	001_4_5_15_2024-03-10.jpg	Healthy	userdata/1/1/001_4_5_15_2024-03-10.jpg	2024-05-09 22:58:01	0.694	0.304	0.002	15	2024-03-10
16	1	001_4_5_16_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_16_2024-03-10.JPG	2024-05-09 22:59:01	0.966	0.0339	0.0001	16	2024-03-10
17	1	001_4_5_17_2024-03-10.jpg	Healthy	userdata/1/1/001_4_5_17_2024-03-10.jpg	2024-05-09 23:00:01	0.966	0.0339	0.0001	17	2024-03-10

Google Cloud Storage



Google Cloud Functions

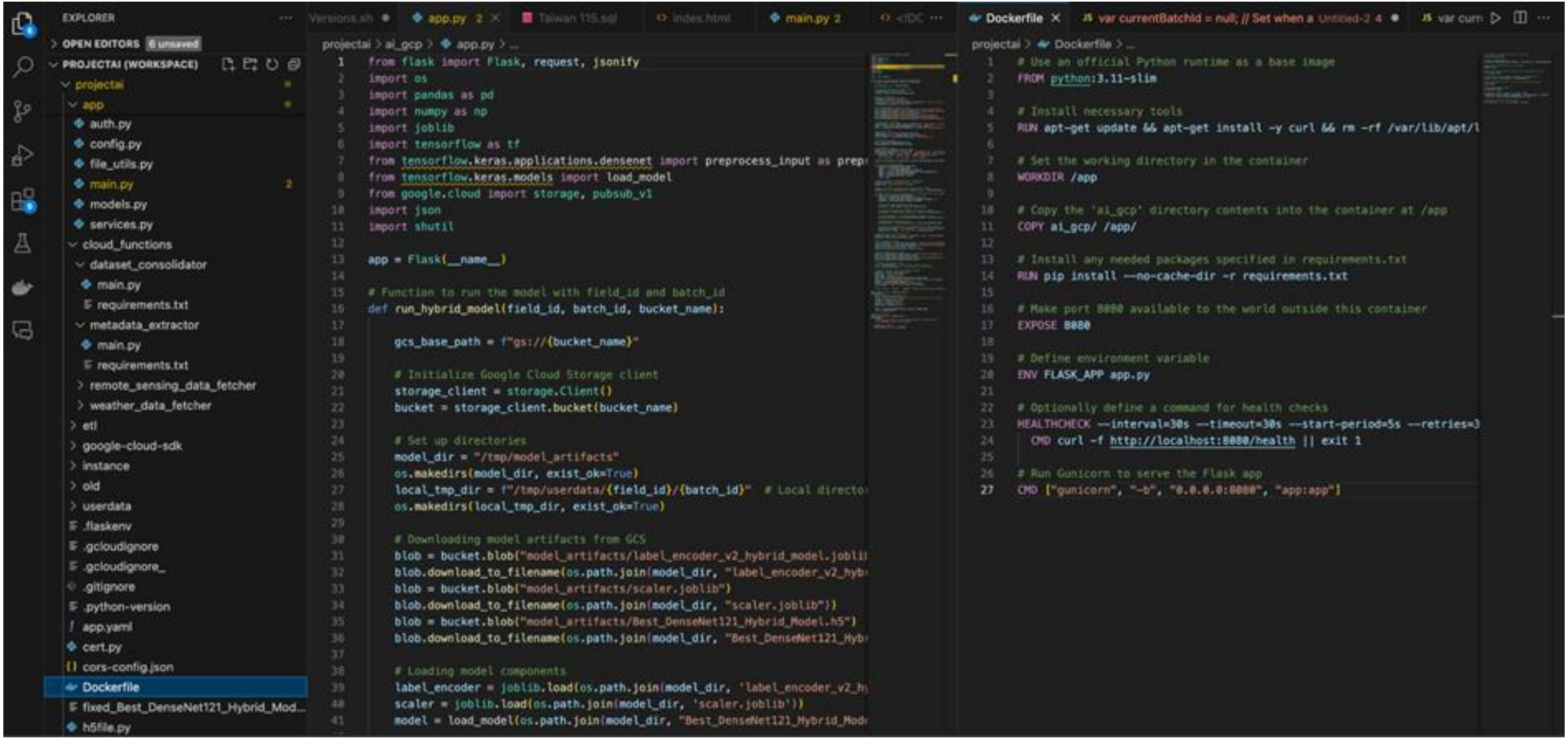
Cloud Functions | Functions

Filter | Filter functions

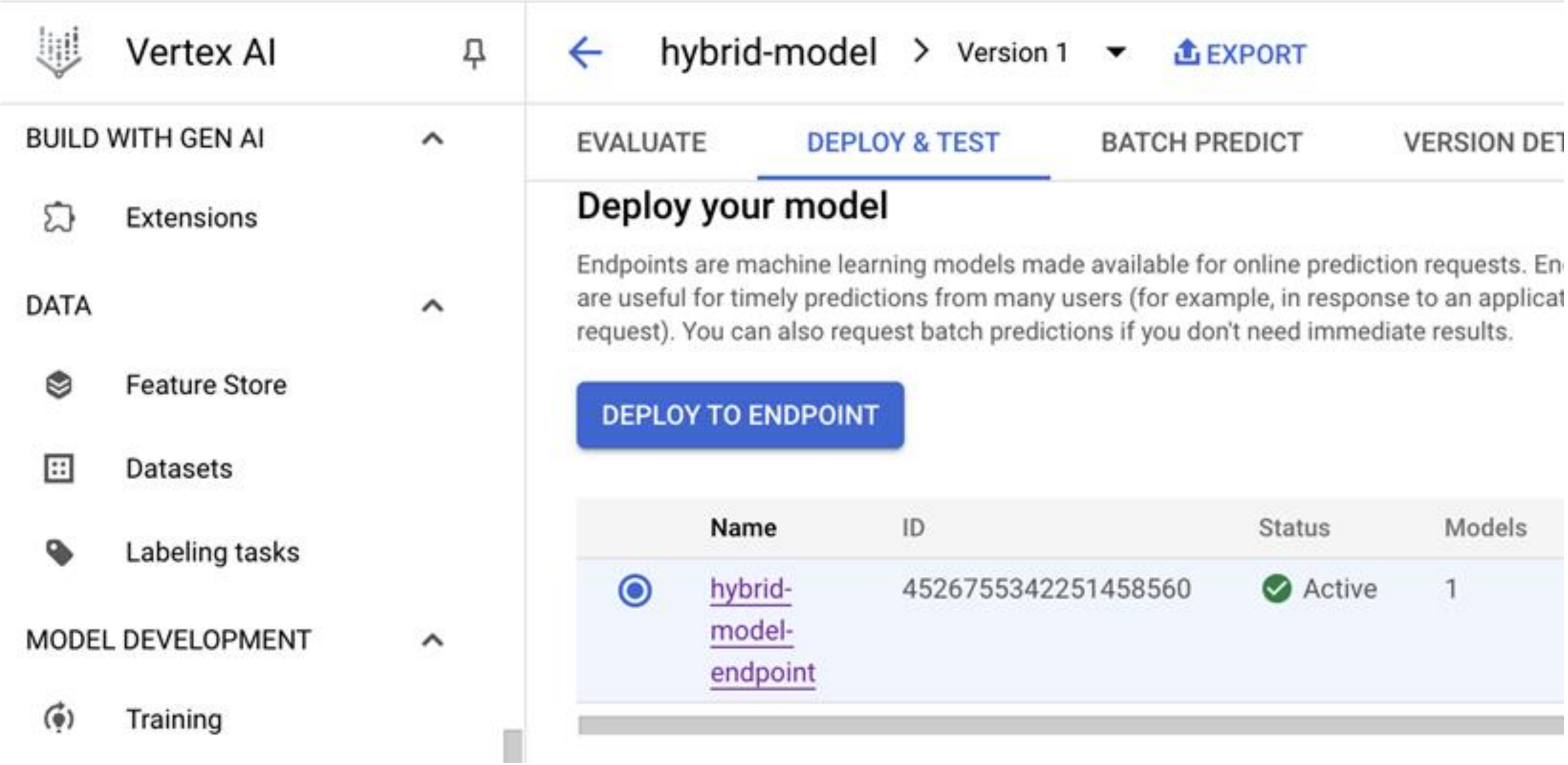
	Environment	Name ↑
✓	1st gen	dataset_consolidator
✓	1st gen	metadata_extractor
✓	1st gen	remote_sensing_data_fetcher
✓	1st gen	weather_data_fetcher

```python
def fetch_remote_sensing_data(event, context):
    """Triggered by the message from metadata_extractor cloud function"""
    print("Raw event data:", event['data'])  # Log the raw base64 message
    pubsub_message = base64.b64decode(event['data']).decode('utf-8')
    print("Decoded message:", pubsub_message)  # Log the decoded string
    message_data = json.loads(pubsub_message)
    print("Message data:", message_data)  # Log the dictionary

    bucket_name = message_data["bucket"]
    field_id = message_data["field_id"]
    batch_id = message_data["batch_id"]

    try:
        # Initialize Earth Engine and Storage Client
        ee.Initialize()
        storage_client = storage.Client()

        # Setup paths and download the metadata CSV to /tmp directory
        bucket = storage_client.bucket(bucket_name)
        local_dir = f"/tmp/{field_id}/{batch_id}/"
        os.makedirs(local_dir, exist_ok=True)
        local_path = os.path.join(local_dir, "image_metadata.csv")
        blob = bucket.blob(f"userdata/{field_id}/{batch_id}/image_metadata.csv")
        blob.download_to_filename(local_path)

        # Read the CSV and process data
        df = pd.read_csv(local_path)

        # List which contains tuples which contain the Latitude, Longitude, and D
        coordinates_and_dates = []

        for _, row in df.iterrows():
            lat = row['Latitude']
```

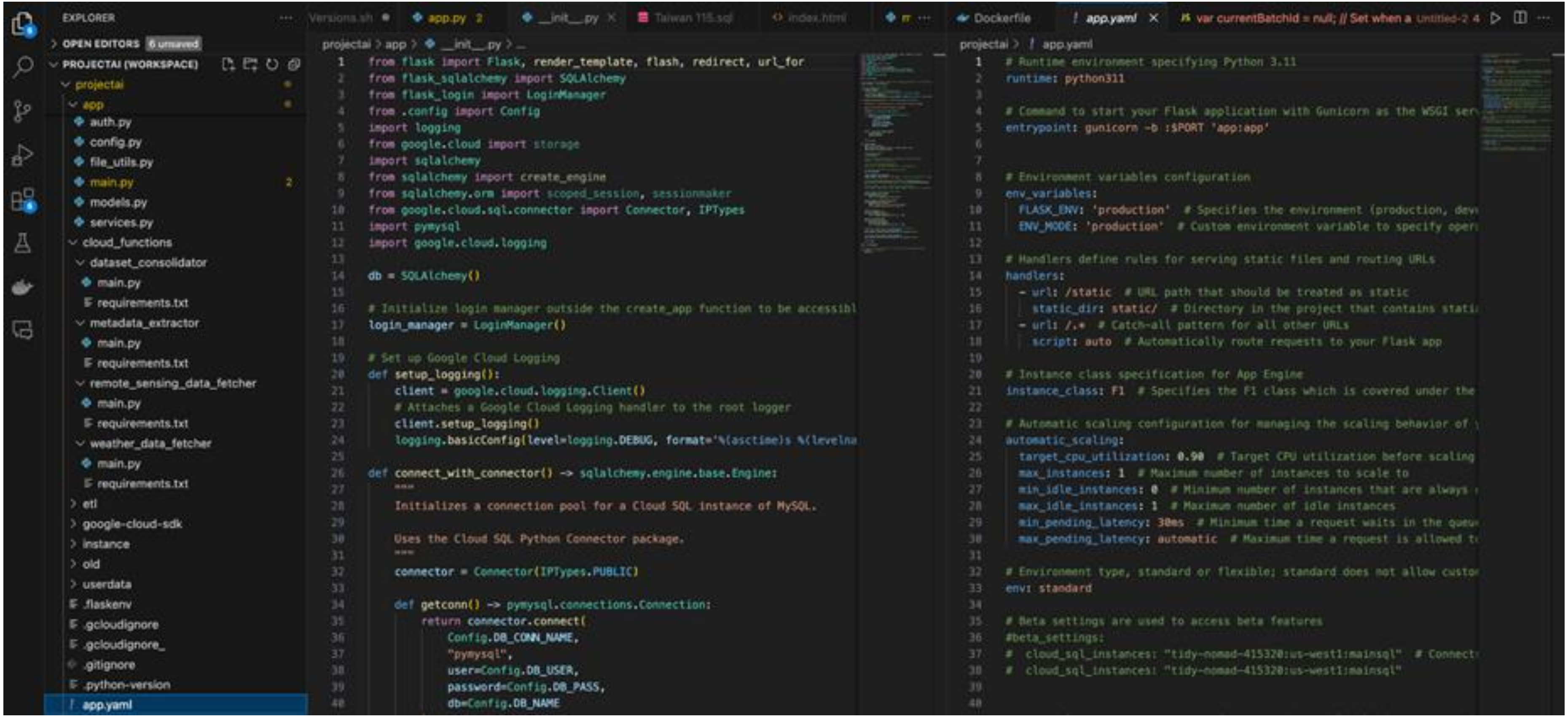
Docker container for hybrid model



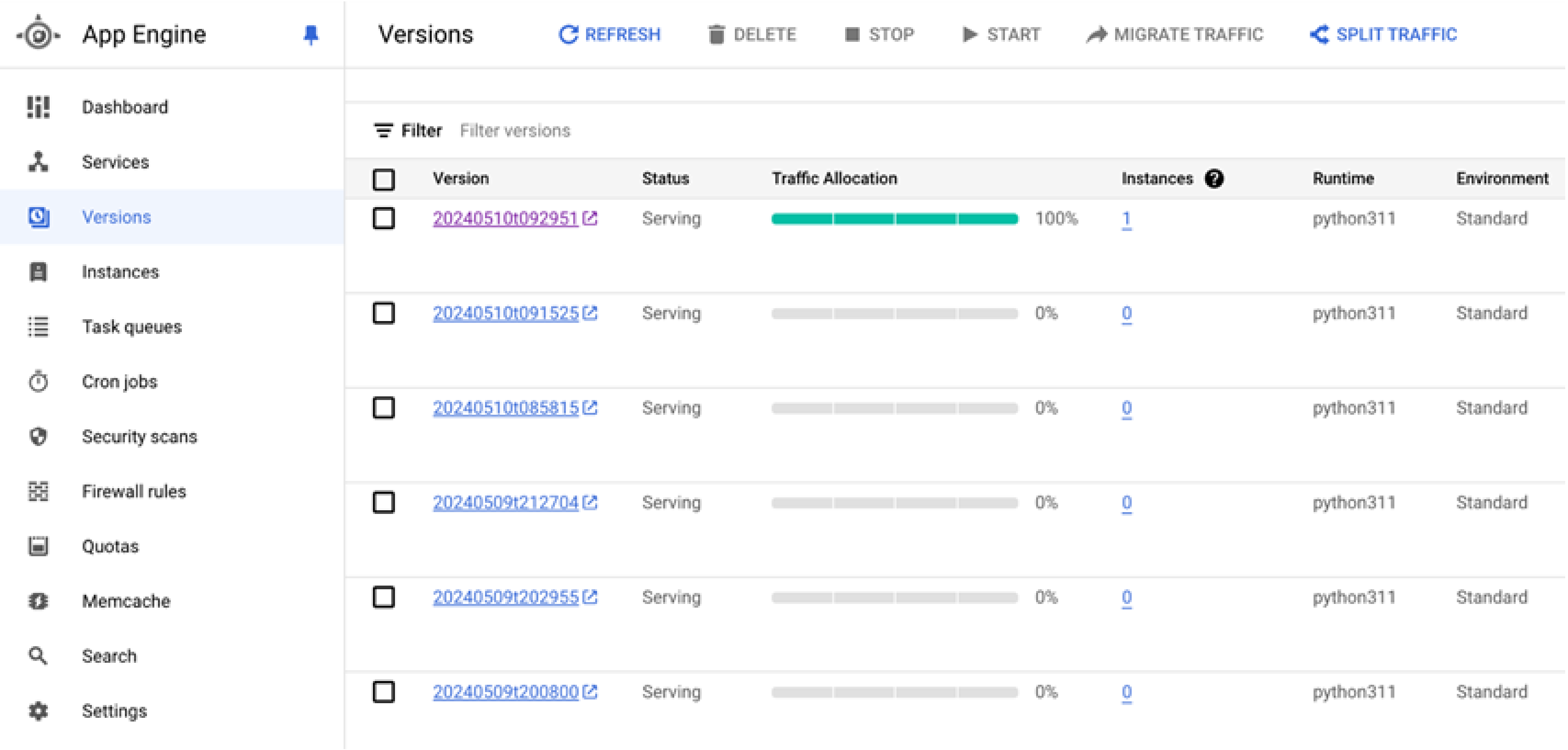
Google Vertex AI



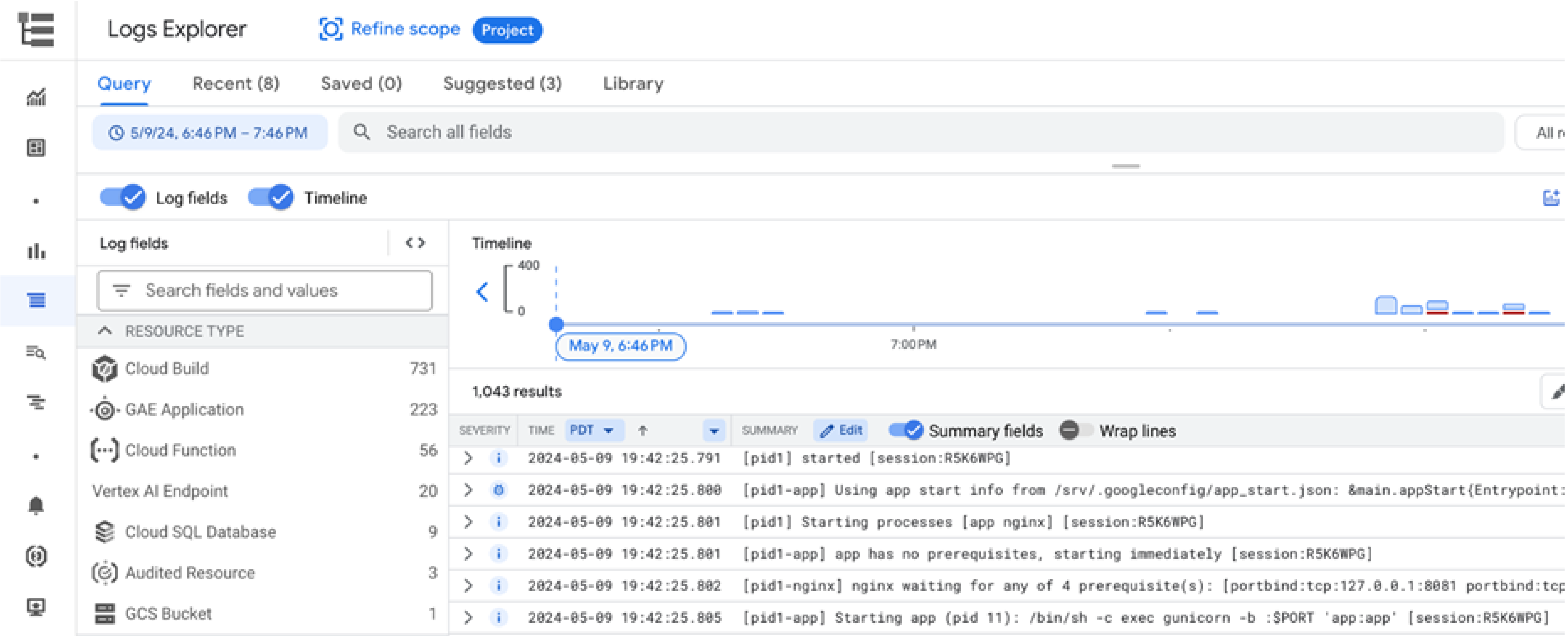
YAML deployment configuration for App Engine



Google App Engine

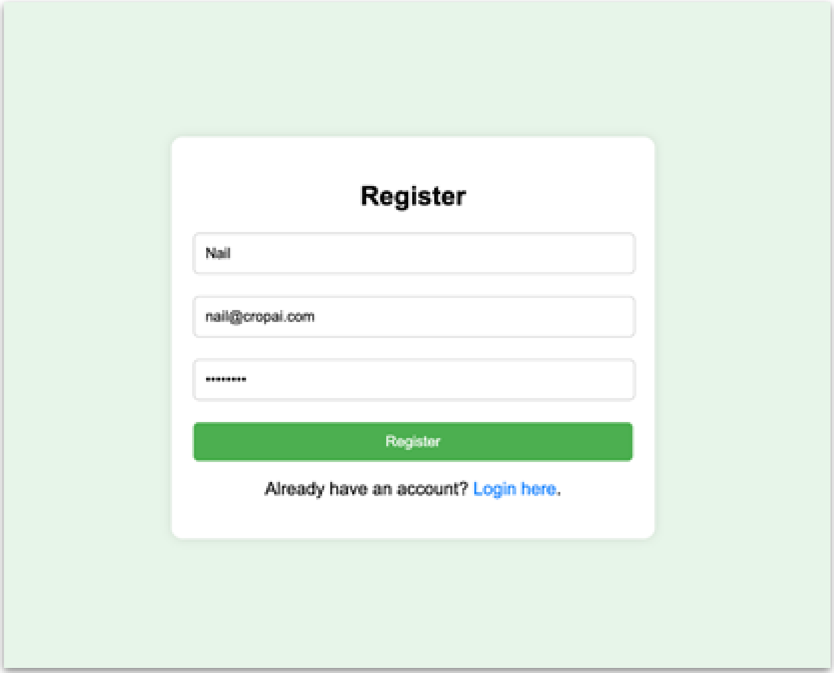


Extensive Logging

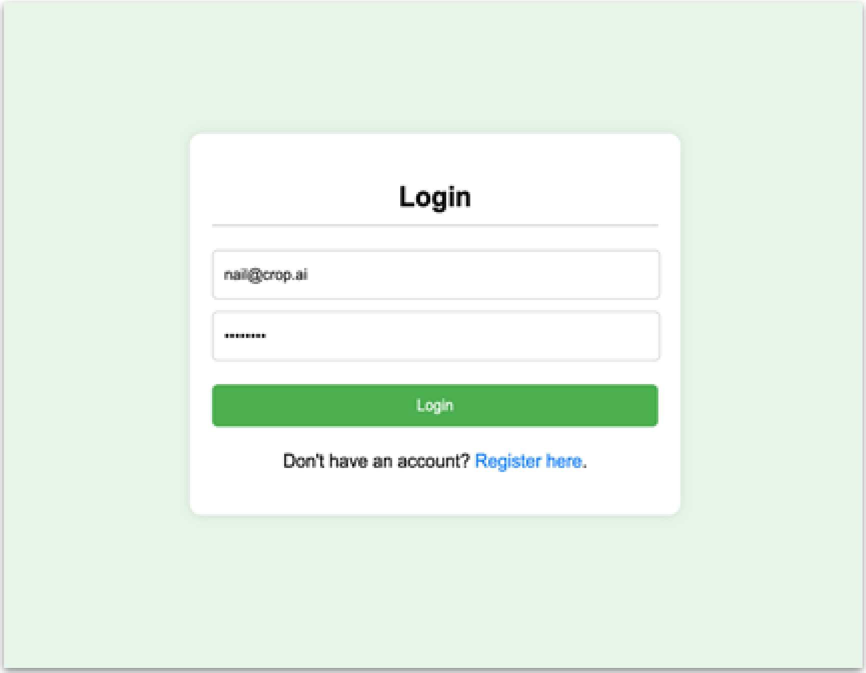


User Interface

Registration Page

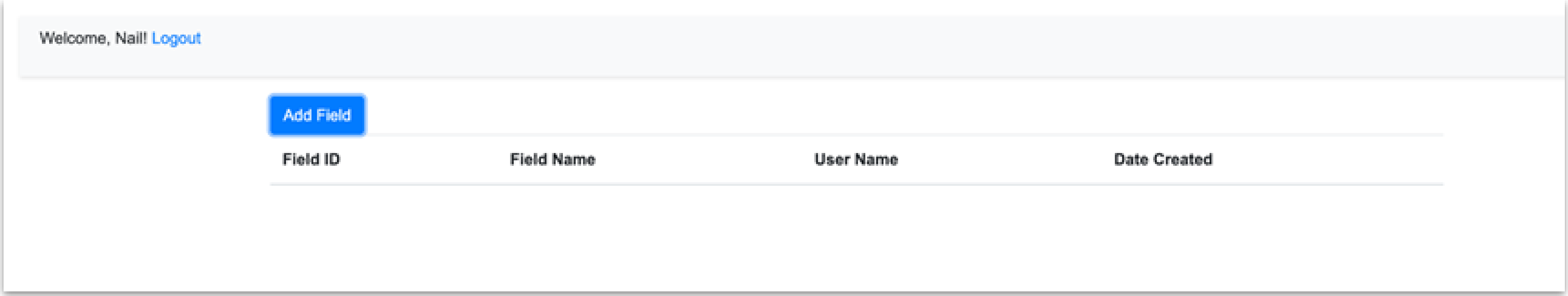


Login Page

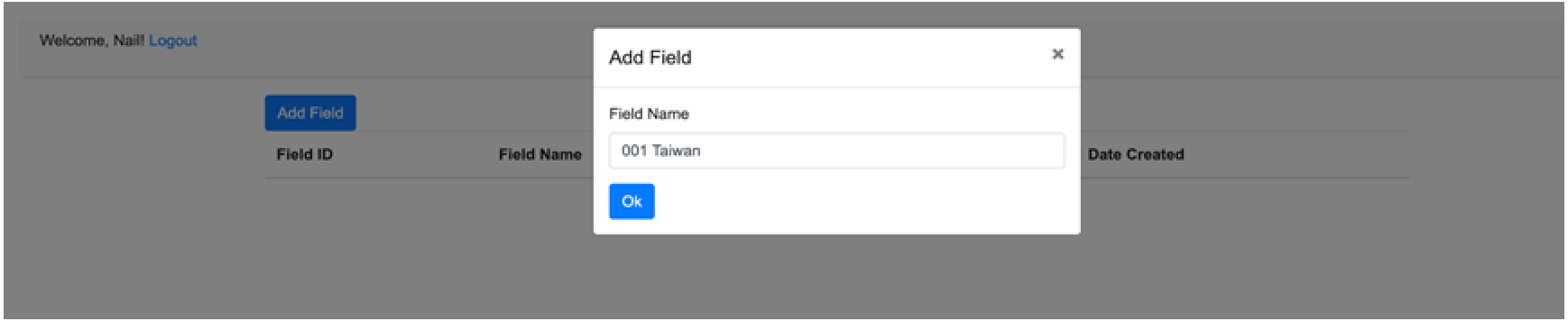


Main Page

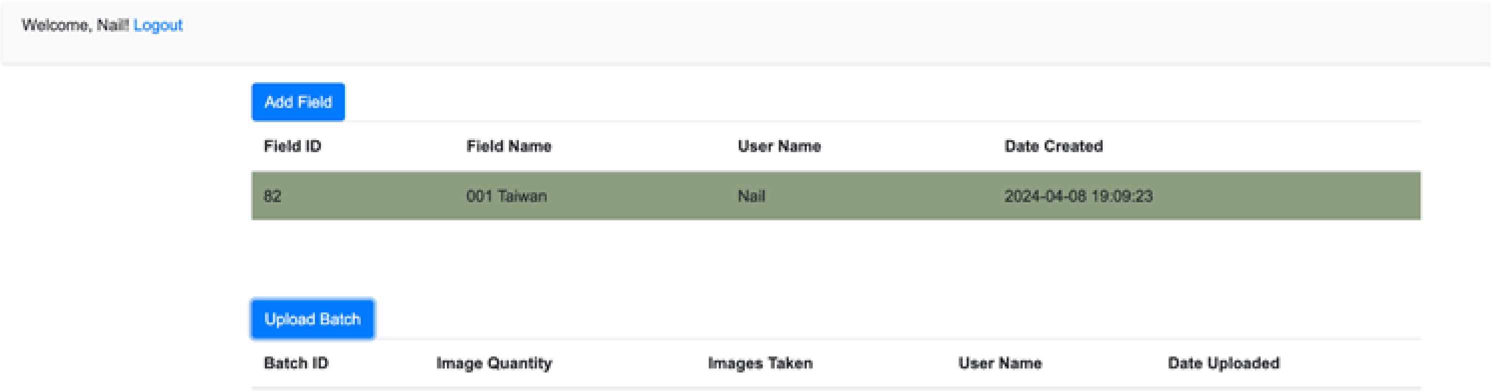
Field Section



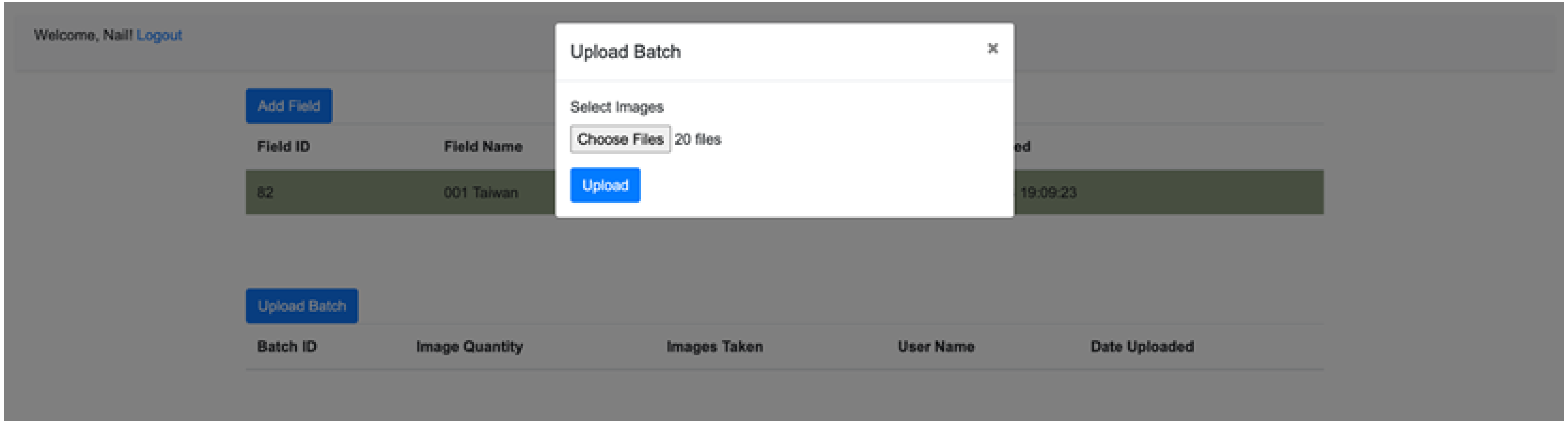
Add Field Form



Batch Section



Upload Batch Form



Data Extraction + Connecting the Model

Data Fetching in Progress

Upload Batch

Batch ID	Image Quantity	Images Taken	User Name	Date Uploaded
95	20	2024-02-01	Nail	2024-04-08 19:26:00

Images

Image ID	File Name	Label	Healthy	Rice Blast	Brown Spot
795	001_4_5_1_2024-02-01.JPG	no	-	-	-
796	001_4_5_2_2024-02-01.JPG	no	-	-	-
797	001_4_5_3_2024-02-01.JPG	no	-	-	-
798	001_4_5_4_2024-02-01.JPG	no	-	-	-

After a Batch has been uploaded, the Data Pipeline runs the Cloud Function to extract metadata from the batch images.

Using the extracted metadata other Cloud Functions fetch weather and remote sensing data.

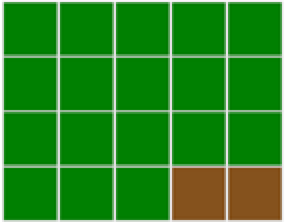
Finally, the consolidated dataset is prepared and along with the images passed to the hybrid model for predictions.

Classification Result

Healthy Brown Spot Leaf Blast

Upload Batch

Batch ID	Image Quantity	Images Taken	User Name	Date Uploaded
100	20	2024-02-01	Nail	2024-04-08 19:48:41



Images

Image ID	File Name	Label	Healthy	Rice Blast	Brown Spot
911	001_4_5_17_2024-02-01.jpg	Healthy	0.966	0.0339	0.0001
912	001_4_5_18_2024-02-01.jpg	Healthy	0.966	0.0339	0.0001
913	001_4_5_19_2024-02-01.jpg	Brown Spot	0	0.0019	0.998
914	001_4_5_20_2024-02-01.jpg	Brown Spot	0	0.0012	0.9988

Disease Progression Results

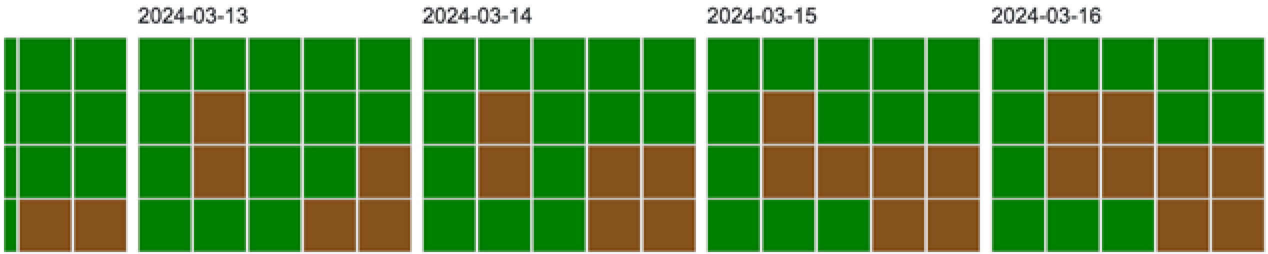
Healthy Brown Spot Leaf Blast

Welcome, Nail! Logout

Add Field

Field ID	Field Name	User Name	Date Created
1	Taiwan 115	Jason	2024-05-08 22:28:28

Disease progression



Upload Batch

Batch ID	Image Quantity	Images Taken	User Name	Date Uploaded
1	20	2024-03-10	Jason	2024-05-08 22:30:30
2	20	2024-03-11	Jason	2024-05-09 22:31:59
3	20	2024-03-12	Jason	2024-05-09 22:32:32
4	20	2024-03-13	Jason	2024-05-09 22:32:58


Demo Video

Thank You!

References found in report