Macquarie University

Sydney, Australia

Evaluating the Efficacy of Local vs. LLM Models in Plant Disease Prediction Across Diverse Species

Phuc Tuong Ngo

Supervised by
Dr. Qiongkai Xu, Dr. Xiaohan Yu

May 2025

Abstract

This research aims to benchmark the performance of local machine learning model against Large Language model (LLM) in regard to image classification for plant disease detection.

The accuracy of the model is critical to enhance agriculture and promote sustainability. Given the importance of this, the case study utilises a comparative analysis to evaluate both models across a variety of plant species and sub-species. By incorporating a subset of Plant Leaf Diseases Training Dataset, we have carried out extensive testing on both local and LLM, measuring the performance in terms of accuracy and precision across various benchmarks.

The findings reveal noticeable differences with local model demonstrating outstanding performance in prediction accuracy. These result suggests directions for future research in further local model optimisation, tailoring to agricultural needs. This study contributes to the application of advanced AI techniques in agricultural industry and proposal for viable research directions for practical application of these technologies in real-world settings.

Introduction

The application of machine learning, specifically in image classification has revolutionised disease detection as suggested by (1). However, the usage of such technology mobilely and accurately, especially in the agricultural environment has presented conundrums. This research paper aims to deliver a comparison in the performance of a fine tuned local model, in comparison to advanced large language models in terms of image classification for plant disease detection.

- **Research Question:** How do local machine learning models perform in comparison to Large Language Models in regards of accuracy in plant disease detection across various plant's species and sub-species.
- **Significance:** The importance of this research grows beyond the pursuit of academia, focusing to provide real-world benefit for the agricultural industry.

Methodology

0.1 Data Collection

The dataset was sourced from Kaggle's "Plant Leaf Disease Training Dataset, which is a combination of a variety of different sub-datasets such as Cassava Leaf Disease Dataset, Rice Leaf Disease Images, PlantVillage, Potato Leaf Disease. This aggregation dataset provides a diverse variety of plant species such as apple, orange, peach, potato, each with specific disease such as black rot, rust scab, bacterial spot as well as healthy specimens.

0.2 Data Selection

The whole dataset is around too large to be benchmarked by LLM. To ensure a balanced, unbiased representation, a random selection script was written to choose 70 images from each sub-category for each plant species. This approach will ensure that each plant species and its respective sub-category are equally represented. Therefore providing a balanced subset of data for model benchamarking.

```
import os
import random
import shutil

# Path configuration
SOURCE_DIR = "path/to/full/dataset"
DEST_DIR = "path/to/subset"
os.makedirs(DEST_DIR, exist_ok=True)

# Randomly select and copy images
for category in os.listdir(SOURCE_DIR):
    images = os.listdir(os.path.join(SOURCE_DIR, category))
    selected_images = random.sample(images, 70)
    for image in selected_images:
        src = os.path.join(SOURCE_DIR, category, image)
        dst = os.path.join(DEST_DIR, category, image)
        shutil.copy(src, dst)
```

0.3 Data Collection

The dataset was sourced from Kaggle's "Plant Leaf Disease Training Dataset, which is a combination of a variety of different sub-datasets such as Cassava Leaf Disease Dataset, Rice Leaf Disease Images, PlantVillage, Potato Leaf Disease. This aggregation dataset provides a diverse variety of plant species such as apple, orange, peach, potato, each with specific disease such as black rot, rust scab, bacterial spot as well as healthy specimens.

0.4 Data Selection

The whole dataset is around too large to be benchmarked by LLM. To ensure a balanced, unbiased representation, a random selection script was written to choose 70 images from each sub-category for each plant species. This approach will ensure that each plant species and its respective sub-category are equally represented. Therefore providing a balanced subset of data for model benchamarking.

```
import os
import random
import shutil

# Path configuration
SOURCE_DIR = "path/to/full/dataset"
DEST_DIR = "path/to/subset"
os.makedirs(DEST_DIR, exist_ok=True)

# Randomly select and copy images
for category in os.listdir(SOURCE_DIR):
    images = os.listdir(os.path.join(SOURCE_DIR, category))
    selected_images = random.sample(images, 70)
    for image in selected_images:
        src = os.path.join(SOURCE_DIR, category, image)
        dst = os.path.join(DEST_DIR, category, image)
        shutil.copy(src, dst)
```

1 Model Selection

This study used two models to analyse the correctness of image classification techniques for plant disease detection: Local Machine Learning Model and Large Language Model (LLM).

1.0.1 Local Machine Learning Model: PlantDiseaseDetectorVit2

The local model selected for this study was the ”PlantDiseaseDetectorVit2” from (2), fined-tuned for plant disease detection by Abhiram and hosted on Hugging Face. The base model is vit-base-patch16-224, an open-source Vision Transformer pre-trained on ImageNet-21k, in a supervised environment from (3).

The Model works by dividing images into patches, each patch is linearly embedded into a fixed-size feature vector which converts the image into a sequence of vectors. A classification token is inserted at the beginning of each sequence which serves as a meta data providing additional information about the image. Positional Embeddings are added to each patch vector as well as the token to help the model understands the spatiality of the patches relative to the original image. The sequence of the embedded patches are passed through a series of encoder layers, each with a self-attention mechanism and feed-forward networks, which allowes the model to learn patterns within the data. In essence, ViT treat the images as patches, similarly to how sentences are tokenised. The Positional Embedding strikes similarity in how words are processed in NLP.

1.0.2 Large Language Model: Gemini 2.0 Flash

Gemini 2.0 Flash, free tier was selected for its robust performance in handling complex data as well as its vision capabilities.

2 Model Benchmarking

The evaluation of the models was made using a python script that automatically runs the model and record the answer. The results were collected and saved to a CSV file for further analysis.

The prompt for Gemini was engineered to limit its response to a set of pre-defined labels, which are the labels of the images.

2.0.1 Local Machine Learning Model: PlantDiseaseDetectorVit2

```
def benchMarkLocal():
count = 0
results = []
for folder in os.listdir(SUBSET_DIR):
    folder_path = os.path.join(SUBSET_DIR, folder)
    for img_file in os.listdir(folder_path):
        img_path = os.path.join(folder_path, img_file)
        base64_img = encode_image_to_base64(img_path)
        predicted = predict(base64_img)
        print("finish img", count)
        count = count + 1
        results.append([folder, img_file, predicted["class"]])
```

```
csv_path = os.path.join(CURRENT_DIR, "benchmark_results.csv")
with open(csv_path, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["True Label", "Filename", "Predicted Label"]) # CSV header
    writer.writerows(results)
```

2.0.2 Large Language Model: Gemini 2.0 Flash

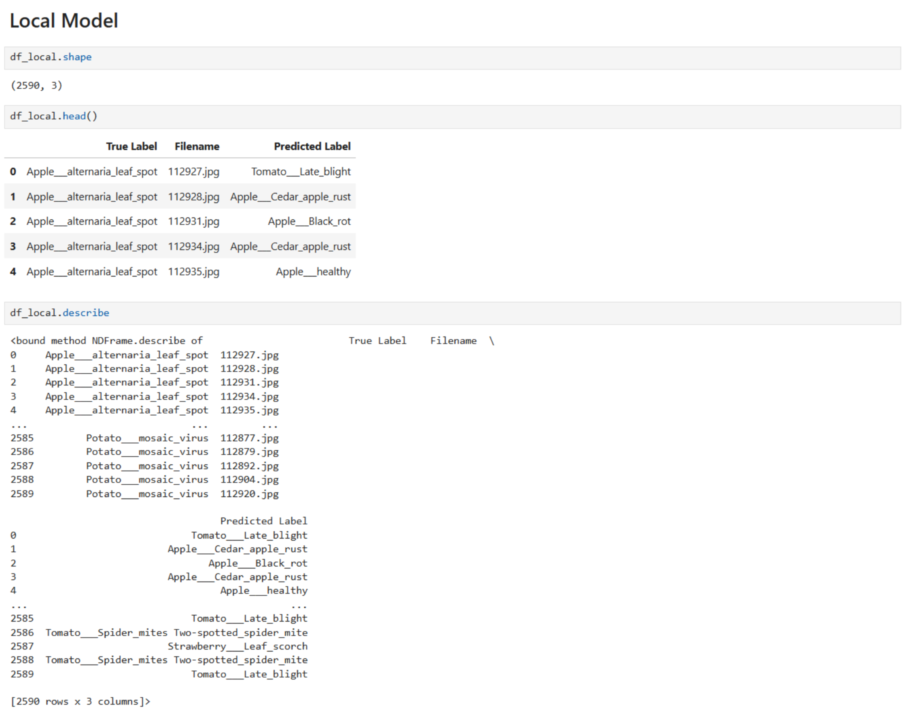
```
def benchMarkGemini():
count = 0
results = []
for folder in os.listdir(SUBSET_DIR):
    folder_path = os.path.join(SUBSET_DIR, folder)
    for img_file in os.listdir(folder_path):
        img_path = os.path.join(folder_path, img_file)
        image = PIL.Image.open(img_path)
        predicted_text = "ERROR: No response"
        retry = 10
        while retry > 0:
            try:
                response = model.generate_content(["You are a plant disease expert.
                    Diagnose this plant and respond only with the exact category
                    name from the list below: ['Apple___alternaria_leaf_spot',
                    'Apple___black_rot', 'Apple___brown_spot', 'Apple___gray_spot',
                    'Apple___healthy', 'Apple___rust', 'Apple___scab',
                    'Bell_pepper___bacterial_spot', 'Bell_pepper___healthy',
                    'Blueberry___healthy', 'Cassava___bacterial_blight',
                    'Cassava___brown_streak_disease', 'Cassava___green_mottle',
                    'Cassava___healthy', 'Cassava___mosaic_disease',
                    'Cherry___healthy', 'Cherry___powdery_mildew',
                    'Coffee___healthy', 'Coffee___red_spider_mite',
                    'Coffee___rust', 'Corn___common_rust', 'Corn___gray_leaf_spot',
                    'Corn___healthy', 'Corn___northern_leaf_blight',
                    'Grape___black_measles', 'Grape___black_rot',
                    'Grape___healthy', 'Grape___Leaf_blight',
                    'Orange___citrus_greening', 'Peach___bacterial_spot',
                    'Peach___healthy', 'Potato___bacterial_wilt',
                    'Potato___early_blight', 'Potato___healthy',
                    'Potato___late_blight', 'Potato___leafroll_virus',
                    'Potato___mosaic_virus']", image])
                predicted_text = response.text.strip()
                break
            except ResourceExhausted as e:
                retry -= 1
                print("Rate limited. Sleeping for 10 seconds...")
                time.sleep(10)
            except Exception as e:
                predicted_text = f"ERROR: {e}"
                break

        print(f"Finished image {count}: {img_file}")
        count += 1
        results.append([folder, img_file, predicted_text])
        if count % 500 == 0:
            print("Saving intermediate results at count =", count)
            save_results_to_csv(results, f"benchmark_gemini_results_{count}.csv")
    save_results_to_csv(results, "benchmark_gemini_results_final.csv")
```

This section details the analysis of the data collected from Plant Leaf Disease Training Dataset. Every steps from initial data handling to model evaluation is discussed to ensure reproducibility and transparency in research findings.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis was conducted to gain insights into the distribution and characteristics of the data.

2.0.3 Local Machine Learning Model: PlantDiseaseDetectorVit2



Local Model

```
df_local.shape
```

```
(2590, 3)
```

```
df_local.head()
```

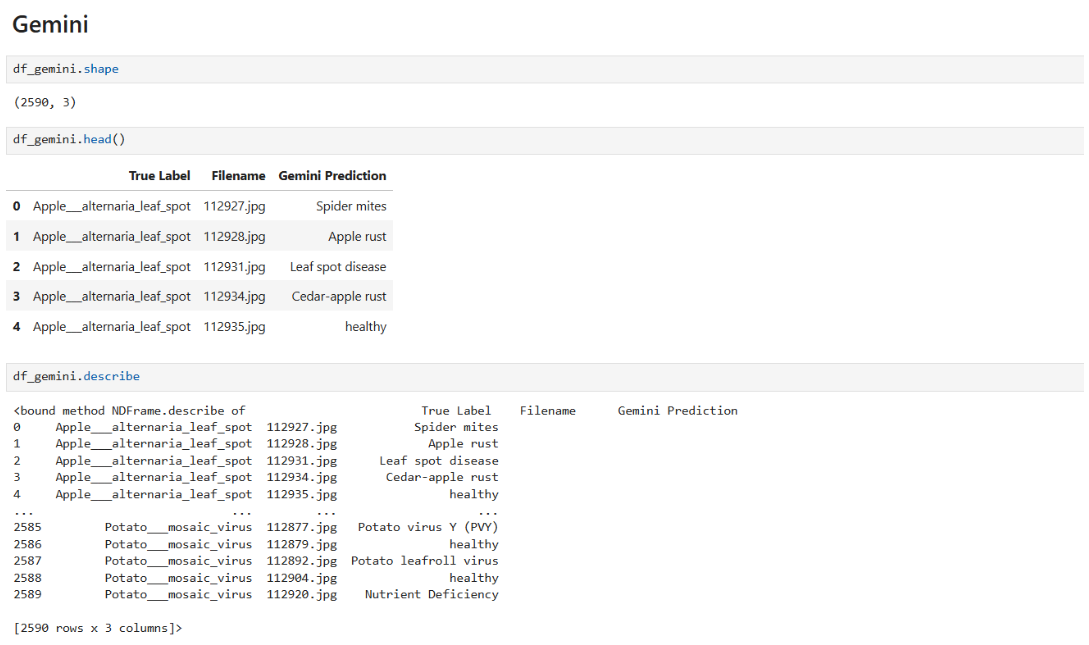
	True Label	Filename	Predicted Label
0	Apple___alternaria_leaf_spot	112927.jpg	Tomato___Late_blight
1	Apple___alternaria_leaf_spot	112928.jpg	Apple___Cedar_apple_rust
2	Apple___alternaria_leaf_spot	112931.jpg	Apple___Black_rot
3	Apple___alternaria_leaf_spot	112934.jpg	Apple___Cedar_apple_rust
4	Apple___alternaria_leaf_spot	112935.jpg	Apple___healthy

```
df_local.describe
```

```
<bound method NDFrame.describe of                         True Label    Filename  \
0     Apple___alternaria_leaf_spot  112927.jpg
1     Apple___alternaria_leaf_spot  112928.jpg
2     Apple___alternaria_leaf_spot  112931.jpg
3     Apple___alternaria_leaf_spot  112934.jpg
4     Apple___alternaria_leaf_spot  112935.jpg
...                            ...         ...
2585         Potato___mosaic_virus  112877.jpg
2586         Potato___mosaic_virus  112879.jpg
2587         Potato___mosaic_virus  112892.jpg
2588         Potato___mosaic_virus  112904.jpg
2589         Potato___mosaic_virus  112920.jpg

                               Predicted Label
0                          Tomato___Late_blight
1                      Apple___Cedar_apple_rust
2                             Apple___Black_rot
3                      Apple___Cedar_apple_rust
4                               Apple___healthy
...                                         ...
2585                       Tomato___Late_blight
2586  Tomato___Spider_mites Two-spotted_spider_mite
2587                     Strawberry___Leaf_scorch
2588  Tomato___Spider_mites Two-spotted_spider_mite
2589                       Tomato___Late_blight

[2590 rows x 3 columns]>
```

Figure 1: Exploratory Data Analysis results showing the description of the local model.

2.0.4 Large Language Model: Gemini 2.0 Flash



Gemini

```
df_gemini.shape
```

```
(2590, 3)
```

```
df_gemini.head()
```

	True Label	Filename	Gemini Prediction
0	Apple__alternaria_leaf_spot	112927.jpg	Spider mites
1	Apple__alternaria_leaf_spot	112928.jpg	Apple rust
2	Apple__alternaria_leaf_spot	112931.jpg	Leaf spot disease
3	Apple__alternaria_leaf_spot	112934.jpg	Cedar-apple rust
4	Apple__alternaria_leaf_spot	112935.jpg	healthy

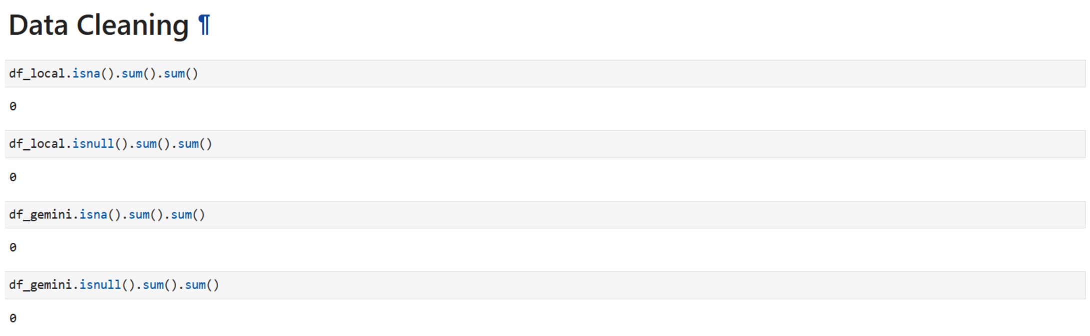
```
df_gemini.describe
```

```
<bound method NDFrame.describe of                   True Label    Filename      Gemini Prediction
0     Apple___alternaria_leaf_spot  112927.jpg         Spider mites
1     Apple___alternaria_leaf_spot  112928.jpg           Apple rust
2     Apple___alternaria_leaf_spot  112931.jpg    Leaf spot disease
3     Apple___alternaria_leaf_spot  112934.jpg     Cedar-apple rust
4     Apple___alternaria_leaf_spot  112935.jpg              healthy
...                            ...         ...                  ...
2585         Potato___mosaic_virus  112877.jpg   Potato virus Y (PVY)
2586         Potato___mosaic_virus  112879.jpg              healthy
2587         Potato___mosaic_virus  112892.jpg  Potato leafroll virus
2588         Potato___mosaic_virus  112904.jpg              healthy
2589         Potato___mosaic_virus  112920.jpg    Nutrient Deficiency

[2590 rows x 3 columns]>
```

Figure 2: Exploratory Data Analysis results showing the description of the LLM.

Data Cleaning

Data cleaning methods were used to clean the data of null, None, NA values before further analysis is conducted. There were no null, none, na value found in the dataset.



Data Cleaning ¶

```
df_local.isna().sum().sum()
```

```
0
```

```
df_local.isnull().sum().sum()
```

```
0
```

```
df_gemini.isna().sum().sum()
```

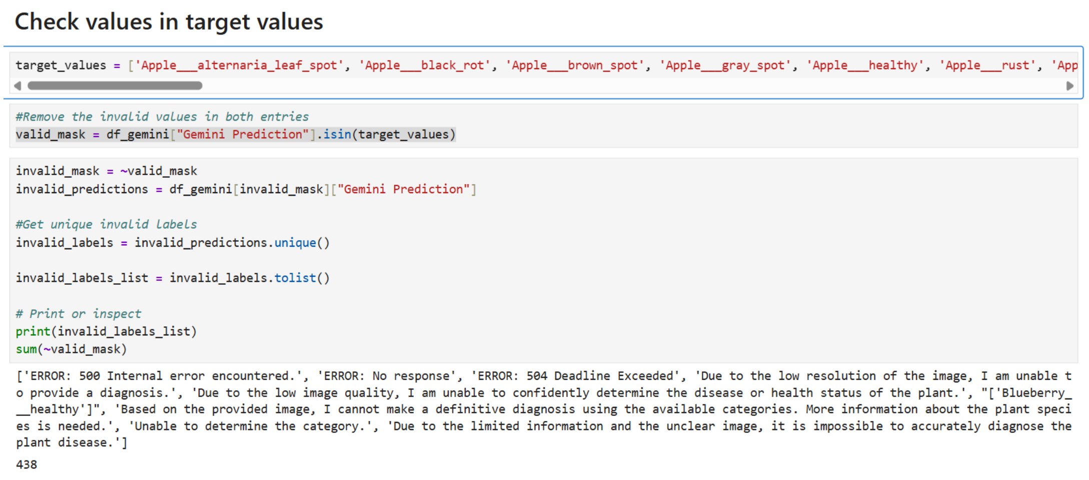
```
0
```

```
df_gemini.isnull().sum().sum()
```

```
0
```

Figure 3: Data Validation Results showing the absence of null, None, or NA values across the entire dataset.

Interestingly, when checked for answers outside the constraints set in the prompt engineer for Gemini, 438 values were found.



Check values in target values

```
target_values = ['Apple___alternaria_leaf_spot', 'Apple___black_rot', 'Apple___brown_spot', 'Apple___gray_spot', 'Apple___healthy', 'Apple___rust', 'App
```

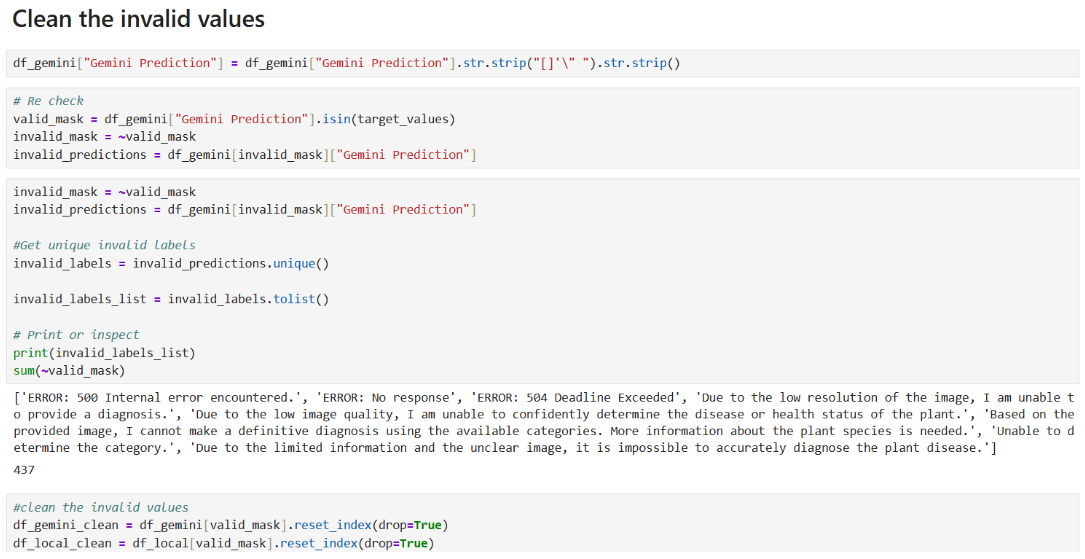
```
#Remove the invalid values in both entries
valid_mask = df_gemini["Gemini Prediction"].isin(target_values)
```

```
invalid_mask = ~valid_mask
invalid_predictions = df_gemini[invalid_mask]["Gemini Prediction"]

#Get unique invalid labels
invalid_labels = invalid_predictions.unique()

invalid_labels_list = invalid_labels.tolist()

# Print or inspect
print(invalid_labels_list)
sum(~valid_mask)
```

```
['ERROR: 500 Internal error encountered.', 'ERROR: No response', 'ERROR: 504 Deadline Exceeded', 'Due to the low resolution of the image, I am unable t
o provide a diagnosis.', 'Due to the low image quality, I am unable to confidently determine the disease or health status of the plant.', "['Blueberry_
__healthy']", 'Based on the provided image, I cannot make a definitive diagnosis using the available categories. More information about the plant speci
es is needed.', 'Unable to determine the category.', 'Due to the limited information and the unclear image, it is impossible to accurately diagnose the
plant disease.']
438
```

Figure 4: Data Validation Results showing the presence of response not in constraint list across the entire dataset.

In the invalid results, there were a result "Blueberry healthy" which was flagged invalid despite in the list. This is due to the invalid check considering absolute similarity and not considering " or [ ] or any other special symbols in the answer. Therefore, that results were filtered out of the invalid to be a valid answer from Gemini. The data is then cleaned.



Clean the invalid values

```
df_gemini["Gemini Prediction"] = df_gemini["Gemini Prediction"].str.strip("[]'\" ").str.strip()
```

```
# Re check
valid_mask = df_gemini["Gemini Prediction"].isin(target_values)
invalid_mask = ~valid_mask
invalid_predictions = df_gemini[invalid_mask]["Gemini Prediction"]
```

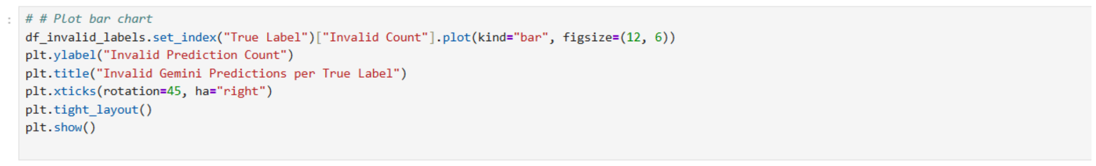
```
invalid_mask = ~valid_mask
invalid_predictions = df_gemini[invalid_mask]["Gemini Prediction"]

#Get unique invalid labels
invalid_labels = invalid_predictions.unique()

invalid_labels_list = invalid_labels.tolist()

# Print or inspect
print(invalid_labels_list)
sum(~valid_mask)
```

```
['ERROR: 500 Internal error encountered.', 'ERROR: No response', 'ERROR: 504 Deadline Exceeded', 'Due to the low resolution of the image, I am unable t
o provide a diagnosis.', 'Due to the low image quality, I am unable to confidently determine the disease or health status of the plant.', 'Based on the
provided image, I cannot make a definitive diagnosis using the available categories. More information about the plant species is needed.', 'Unable to d
etermine the category.', 'Due to the limited information and the unclear image, it is impossible to accurately diagnose the plant disease.']
437
```

```
#clean the invalid values
df_gemini_clean = df_gemini[valid_mask].reset_index(drop=True)
df_local_clean = df_local[valid_mask].reset_index(drop=True)
```

Figure 5: Data Validation Results showing the presence of response not in constraint list across the entire dataset.

When plotted and grouped by the True label, we see an interesting error. Cassava section has some error with noticeably in the cassava healthy label. Cherry performs the worst with the whole cherry section regardless of category got errors. This is incoherent with our previous case study analysis, where Cassava and Cherry performs terribly.



```
# # Plot bar chart
df_invalid_labels.set_index("True Label")["Invalid Count"].plot(kind="bar", figsize=(12, 6))
plt.ylabel("Invalid Prediction Count")
plt.title("Invalid Gemini Predictions per True Label")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```

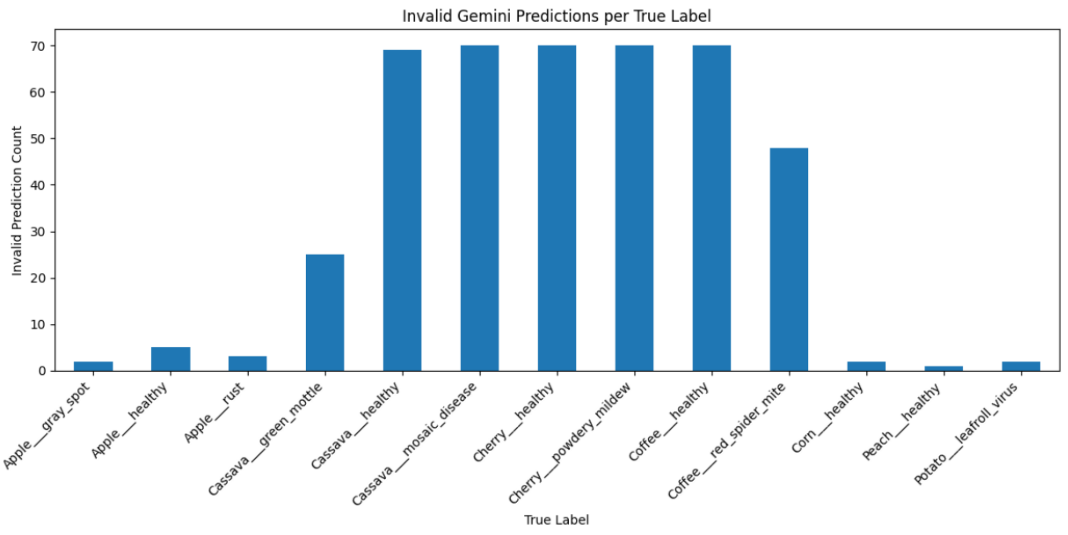


Figure 6: Data Validation Results showing the presence of response not in constraint list across the entire dataset.

Data Transformation

Data transformation is a crucial step in preparing the dataset for effective model evaluation. The data in the model has been re-grouped into streams that fits the purpose of the analysis

```
df_group = df_gemini.copy()
# Split the 'True Label' column and convert to lowercase
df_group[['Plants', 'Disease']] = df_group['True Label'].str.split('___',
    expand=True).map(lambda x: x.lower() if isinstance(x, str) else x)
df_group.pop('True Label')

# Insert at the beginning
df_group.insert(0, 'True Label', df_group.pop('Disease'))
df_group.insert(0, 'Plants', df_group.pop('Plants'))

# To lower case of True Label
df_group["True Label"] = df_group["True Label"].str.replace("_"," ")

# Insert Local prediction to lower case
df_group.insert(len(df_group.columns), "Local Prediction",df_local["Predicted
    Label"].str.split('___', expand=True)[1].str.replace("_"," ").map(lambda x:
    x.lower() if isinstance(x, str) else x))

#Insert Gemini Prediction to lower case
gemini_prediction = df_group.pop('Gemini Prediction')
df_group.insert(len(df_group.columns), "Gemini
    Prediction",gemini_prediction.str.replace("-"," ").map(lambda x: x.lower() if
    isinstance(x, str) else x))
```

```
filename = df_group.pop('Filename')
df_group.insert(len(df_group.columns), 'Filename', filename)
```

3 Analysis

The analysis was conducted by comparison the value whether it was an exact match with the true label.

3.1 Analysis by Plants

While the local performs decent in the last analysis, overall, Gemini models out perform the Local model with the constraint label given. The local model's performance is significantly weaker.

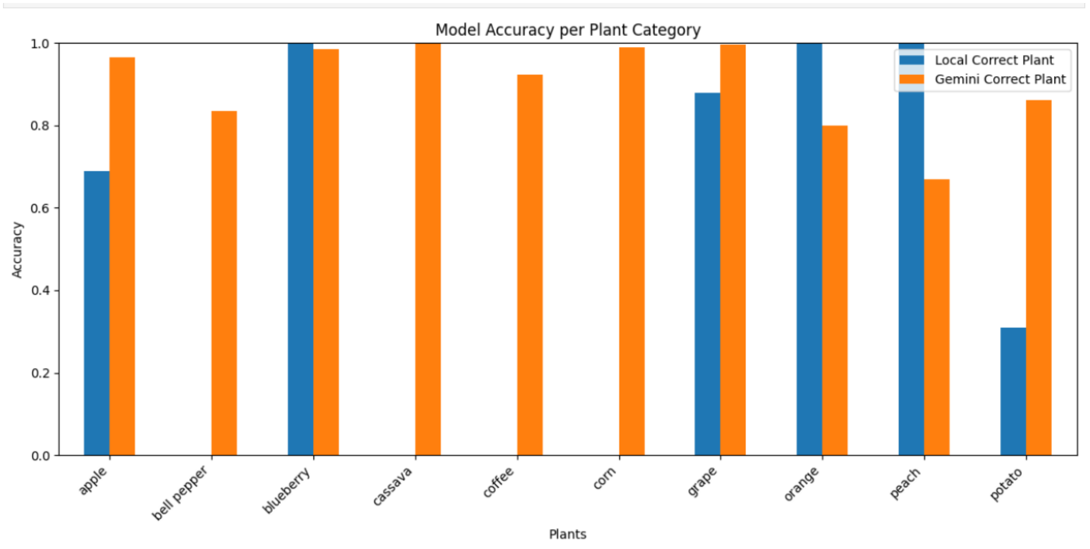


Figure 7: Model Accuracy per Plant Category.

3.1.1 Gemini Plants

In predicting plant's label, Gemini's performance is outstanding, with around 92 percent accuracy to the actual label. Only around 8 percent of labels are wrong.

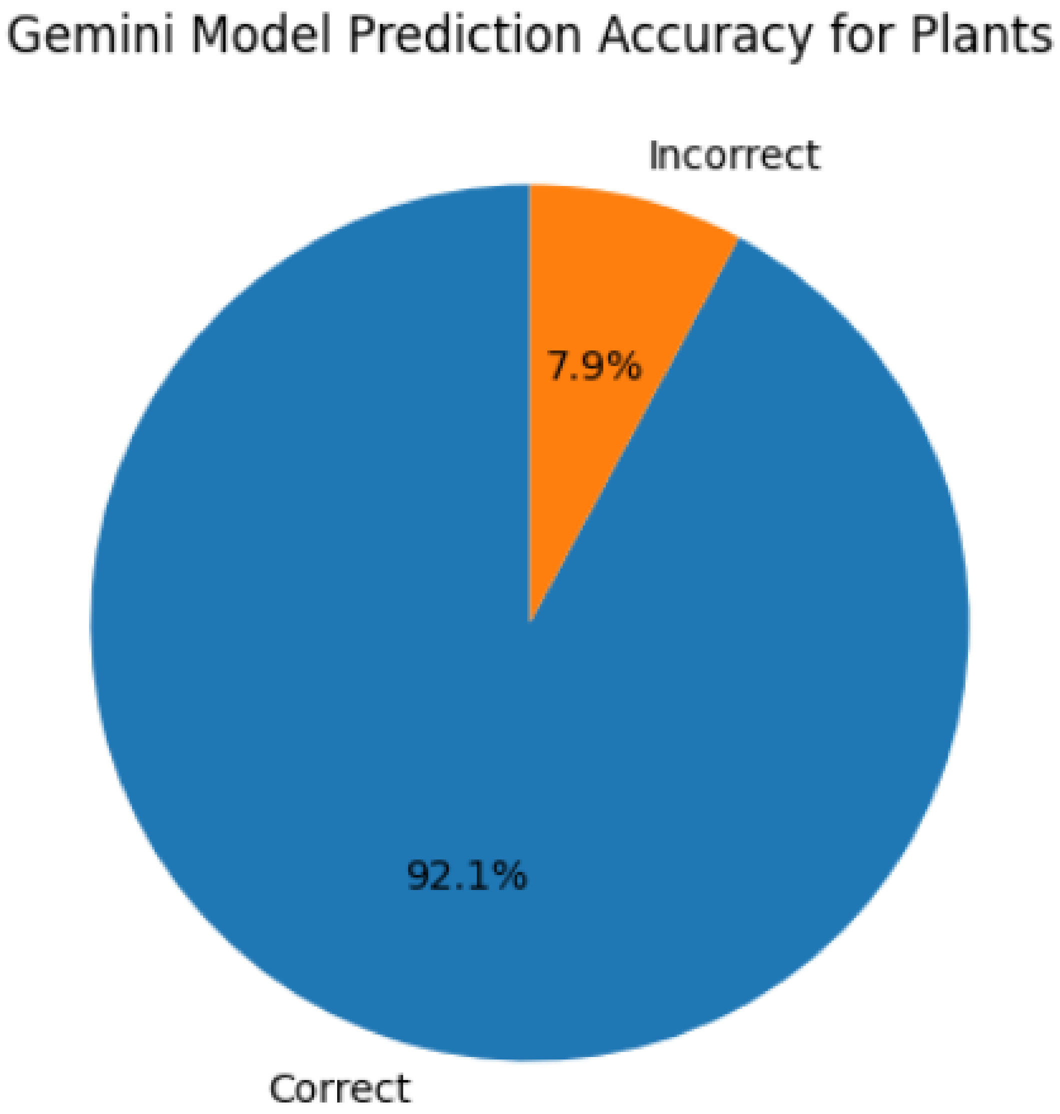


Figure 8: Data Validation Results showing the presence of response not in constraint list across the entire dataset.

3.1.2 Local Plants

The local model's performance is not on par with Gemini with only nearly 46 percent of accuracy and more than half of the predictions are wrong.

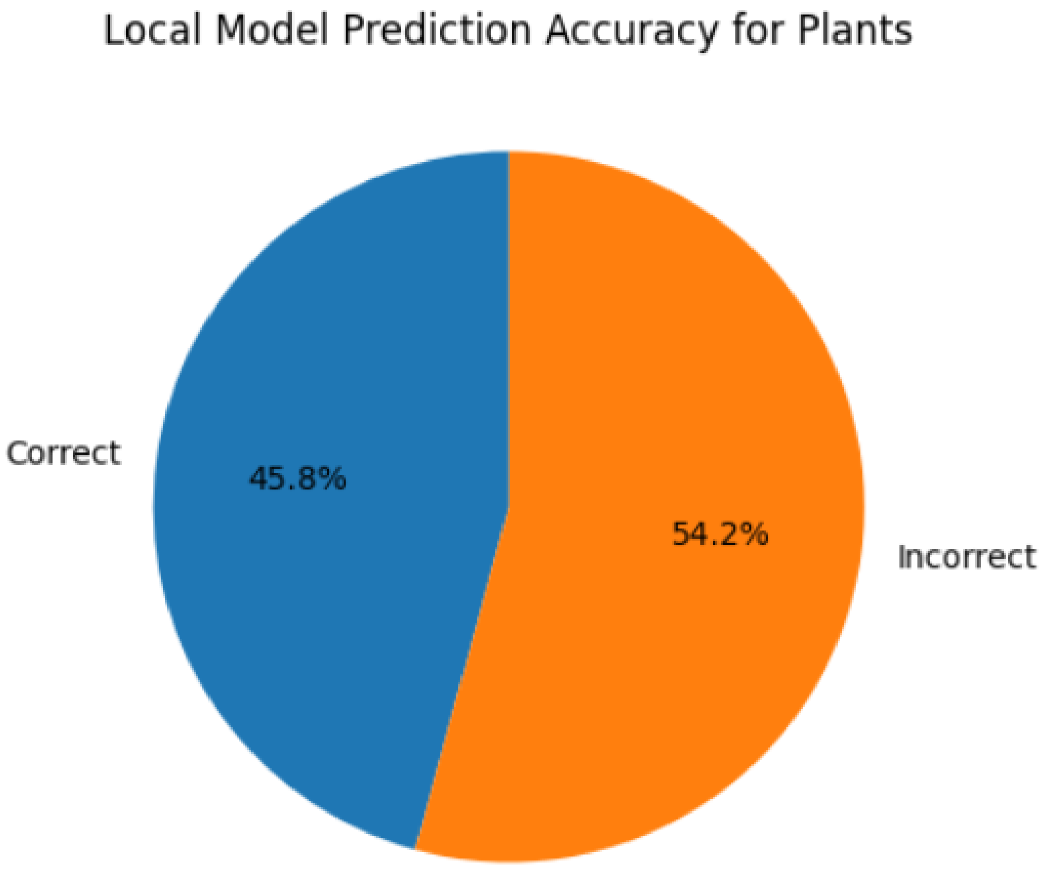


Figure 9: Local Model Prediction Accuracy for Plants.

3.2 Analysis by True Label

In terms of True Label, the disparity between performance is smaller between the two models. Overall, in challenging sections such as Cassava, Coffee, Gemini's performance is low but the Local Model is 0. In other sections such as orange, Gemini performance's gap with the local model is high.

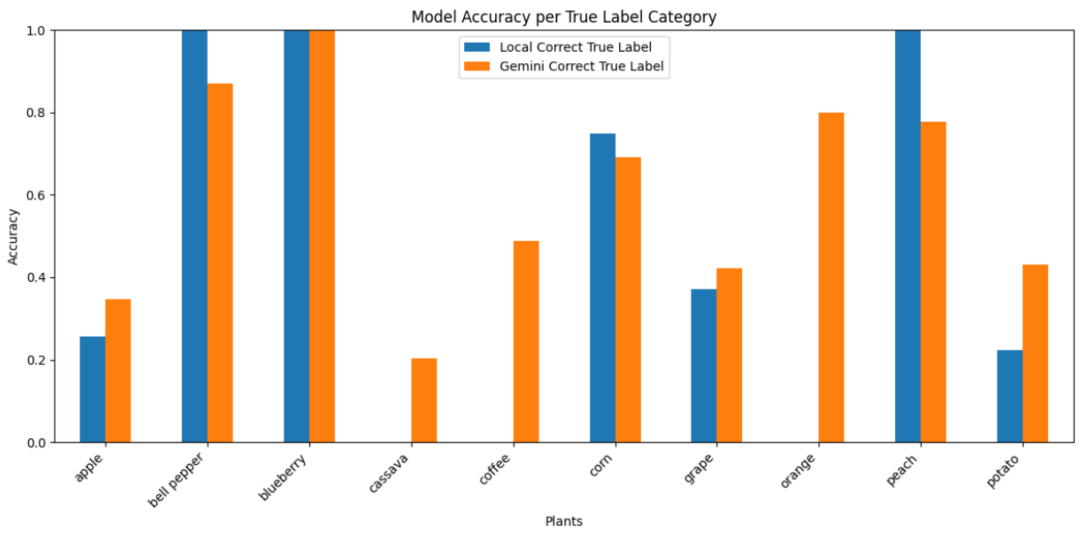


Figure 10: Model Accuracy per True Label Category.

3.2.1 Gemini True Label

Gemini's performance is average, with more than half of the labels correctly predicted.

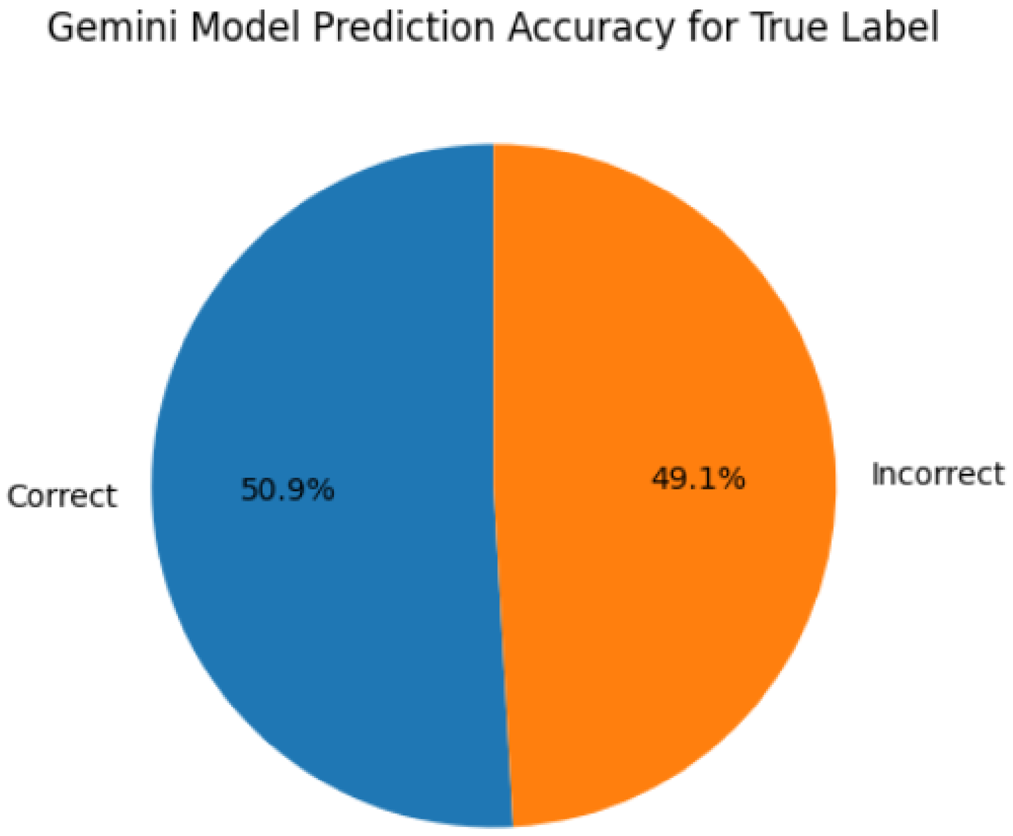


Figure 11: Model Accuracy per True Label Category.

3.2.2 Local True Label

The local model on the other hand has less than half of the predictions incorrect, with only around 40 percent of accuracy.

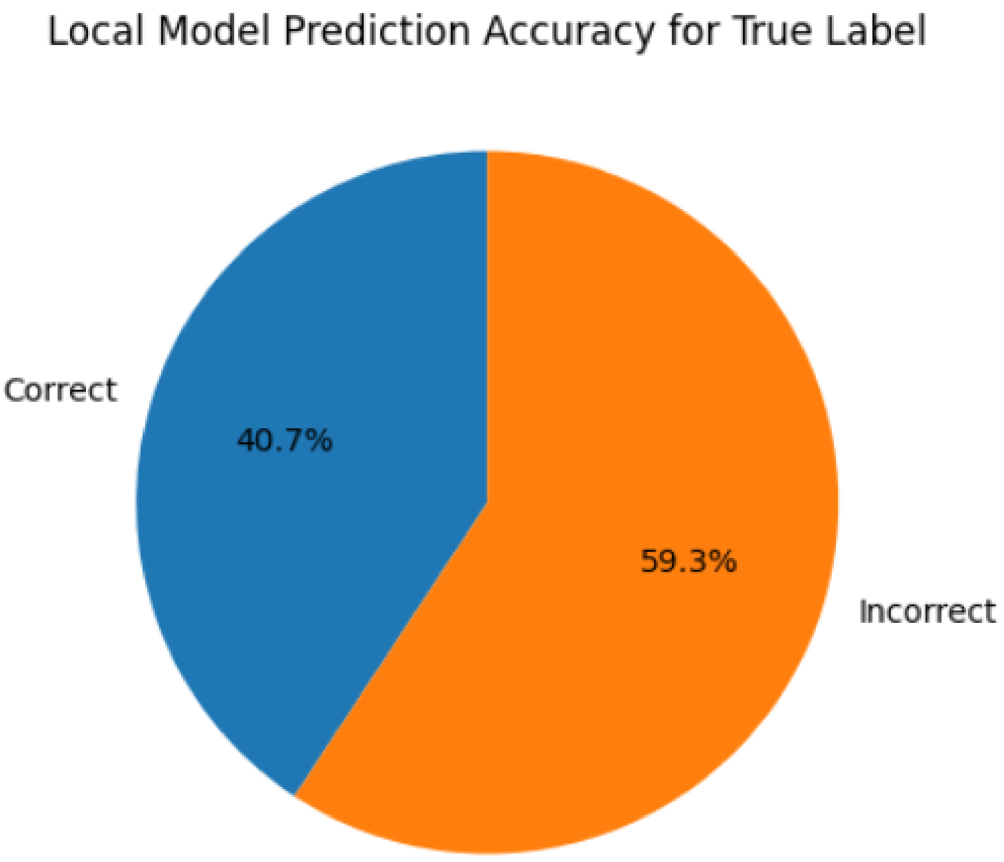


Figure 12: Model Accuracy per True Label Category.

3.3 Analysis by Plants and True Label

Overall$_{plants_{T}rue_{L}abel}$

3.3.1 Gemini Plants and True Label

3.3.2 Local Plants and True Label

4 Discussion

5 Future Work

5.1 Image Classification

While the results favor the local model, the Large Language Model that is used for this study was Gemini 2.0 flash free tier which pose certain limitations in terms of accuracy and efficiency. The paid version of Gemini (2.5 Pro) or alternatives such as GPT-4.5, GPT-4o-latest may yield a different result.

The local model used in this study is only the fine-tuned version of a transformer-vision model, ViT-Base model with only 86M parameters, if the base model had more parameters such as the ViT-Huge with 632M parameters, and fine-tuned with a much more diverse dataset, the results would have been more interesting.

For future studies, consideration for the usage of more recent, advanced convolutional neural networks (CNNs) vision models such as EffectiveNet fine-tuned.

5.2 Evaluation

References

[1] H. Pallathadka, M. Mustafa, D. T. Sanchez, G. Sekhar Sajja, S. Gour, and M. Naved, "Impact of machine learning on management, healthcare and agriculture," *Materials today : proceedings*, vol. 80, pp. 2803–2806, 2023.

[2] Abhiram4, "PlantDiseaseDetectorVit2." `https://huggingface.co/Abhiram4/PlantDiseaseDetectorVit2`, 2023.

[3] Google, "vit-base-patch16-224-in21k." `https://huggingface.co/google/vit-base-patch16-224-in21k`, 2024.