Project Title: Dusty
Group Member: *Adhish Chakravorty, Tsitsi Dube, Jenny Zhang & Johan Lakshmanan*
Type of Project: *Software*
Summary: *A deep learning-based multi-algorithm model aimed at detecting plant-based diseases to reduce agricultural & economic loss.*
Level: *Undergraduate*

Introduction:

Over 40% of agricultural crops are lost to identifiable undetected diseases, resulting in a global annual loss of over $220 billion through food shortages & damaged crops, according to the United Nations' Food & Agriculture Organization. Many crops today are genetically altered and modified to be more immune to pathogens, which has resulted in 'superpests', some of which are more resistant to pesticides and antifungal medications. This has increased the need to be able to identify diseases at a quicker rate to prevent unnecessary damage. With pre-existing solutions being either very computationally expensive & excessively complex, biased towards a dataset & low on accuracy, or just too generic with very few parameters without depth, we decided to address this gap through a novel plant-based disease detection model.

After careful examination of these algorithms, we plan on using Logistic Regression, K-Nearest Neighbors & a Support Vector Machine coupled with a Convolutional Neural Network to predict accuracy to effectively cross-examine results, as opposed to singular, sensitive & large Decision Trees or unrealistic independent Naive Bayes classifiers, allowing for greater accuracy. Moreover, due to regularization & the CNN, we can combat unexpected outliers, high bias & high variance. This fusion of algorithms & precise accuracy prediction set us apart from other solutions, hardware or software.

Background:

Most of the plants we have chosen are common crops, as those have the most readily available and accessible information. Although 80% of plant-based diseases are classified as fungal, the diseases we choose to identify using this software include bacterial and viral infections. Most of these diseases can be identified and categorized by the reaction to the pathogen rather than identification of the pathogen itself, making them much easier to detect & cure. Blemishes, mold, decolorization, rust, and rotting of the leaves are a few common examples of such reactions to pathogens. Through these reactions, we can find the issue with the plant, such as low magnesium or water levels & find the corresponding disease that has matching symptoms.

A Machine Learning Artificial Intelligence Model makes a program more accurate over time without explicit instructions. There are two forms of machine learning we have utilized, namely supervised & unsupervised learning. Supervised Learning operates on inferring common features between pre-labeled data. In contrast, Unsupervised Learning works on inferring common features between unlabeled data by clustering & classifying data into its own labels. Beginning with supervised learning, we used Logistic Regression, which returns either True or False for an input of one of two classes, based on whether its probability of being True or False is above or below a set threshold. We also used the K-Nearest Neighbours Algorithm,

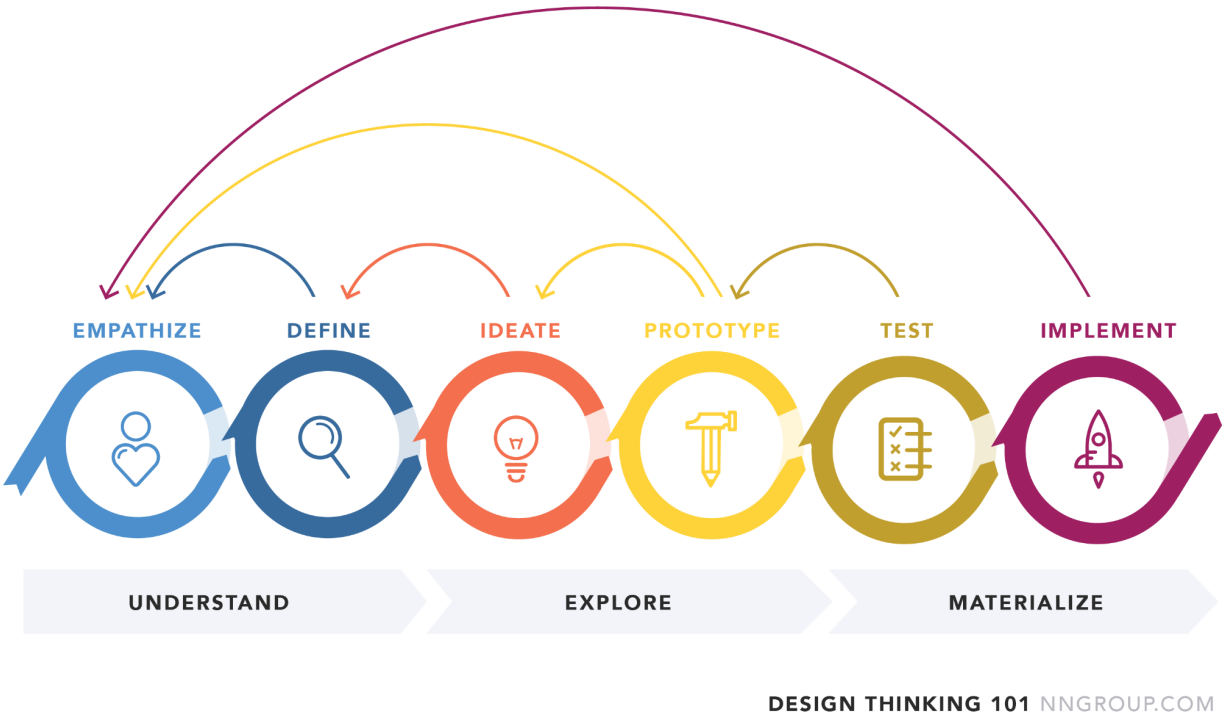
which clusters data that are similar to one another, effectively classifying inputs as a particular disease after going through a cluster of similar symptoms & then a cluster representing each disease. Finally, there is a Convolutional Neural Network, which operates on multiple layers of computation, picking our parameters & comparing the input to the available data, comparing a different set of parameters at each layer. Regularization, a method wherein a high number of parameters increases the cost or lowers the algorithm's accuracy, ensures the minimum number of parameters are used while outputting with the maximum accuracy. As part of unsupervised learning, we used a Support Vector Machine algorithm, which clusters & classifies data similar to the KNN algorithm. SVMs are traditionally used for regression in supervised learning, but newer approaches have begun using it as a form of clustering for unsupervised learning as well, and so have we. Our two-tiered model consists of the SVM, KNN & Logistic Regression, which identifies if the plant is diseased or healthy, followed by the second tier, consisting of the CNN, which determines the plant & disease type. The two-tiered model was taken up due to higher accuracy levels, better accounting for bias & outliers (specifically due to our varied datasets to avoid regional bias), making it more preferable over a single-tier model. After using combinations of algorithms such as Naive Bayes, Linear Discriminant Analysis, and Random Forest & Decision Trees with upto 8 models in 3 tiers, we realized the accuracy began flatlining with the 4 models in 2 tiers model upwards, making it the simplest implementation with the smallest runtime. Details on why we used each model are provided in Methodology 2.

Literature Review (5 sources):

1. Final Plant Dataset: After extensive research, we combined 2 datasets into a singular dataset for our use, containing auto-separated training, cross-validation & testing datasets. The plants included in the datasets are (not exhaustive) apple, bell pepper, blueberry, cherry, corn, peach, potato, raspberry, soybean, squash, strawberry, tomato, orange, and grapes. While in the first dataset, the data are categorized into diseased & healthy to train Tier 1, while the second dataset categorizes the data in terms of the plant & disease type to train Tier 2, which identifies both the plant & the disease type.
2. Signs and symptoms of plant disease: Is it fungal, viral or bacterial? - Jim Isleib: This article lists specific identifications of fungal, bacterial, and viral vegetative disease including, and not limited to, changes in color, shape, and/or functions of plants as it reacts to pathogens, a physical indication of plant-based disease, which can be observed through images. The article presents methods for scientists to track and identify the types of plant disease through identifying the reaction of the plants to the corresponding pathogen, and not necessarily identifying the pathogen itself (as most are not presentable in images/to human eyes).
3. Plant diseases and pests detection based on deep learning: a review - Jun Liu & Xuewei Wang: This review provides a definition of the problem of plant pest detection and a comparison with traditional methods of plant pest detection. According to the different network structures, this study summarizes the recent research on the detection of plant pests and diseases based on deep learning from three aspects: classification network, detection network and segmentation network, and summarizes the advantages and disadvantages of each method. Common datasets are introduced and the performance of existing studies is compared.

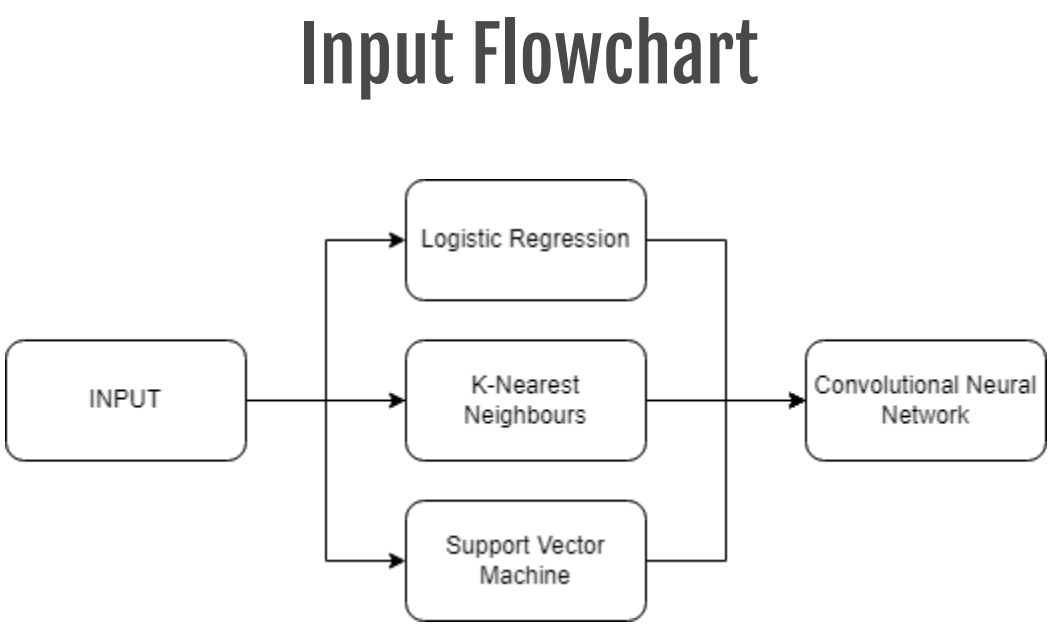
4. Detection of plant leaf diseases using image segmentation and soft computing techniques - Vijai Singh, A.K. Misra: This paper provides a proposed soft computing technique for plant disease identification using image segmentation by applying different types of image processing techniques on the collected images (through digital cameras). The paper provides a step by step approach for the proposed image recognition and segmentation processes, a process of 7 steps: (1)Image acquisition, (2)Image enhancement (extraction of useful information), (3)threshold, (4)infected cluster distinction, (5)obtain useful segmentations for classification, (6) Computing the features using color co-occurrence methodology, and (7)classification of disease.
5. Hyperspec MV.X: This datasheet shows one of the developed hardware product targeting plant disease identifications, where it provides, in detail, of the specific software development steps and functions of the hardware, and also the instructions on how to use it. It also provides a table of the components implemented in the development of such a product with its measurements. This datasheet can serve as an example or guideline in how we can develop our own software while also providing useful information in potential ideas for hardware implementation for this project.

Design Process (Methodology 1):



1. **Problem Identification (July 10):** Funneling down problems & gaps in industries & sectors that cause a measurable impact on society.
2. **Defining The Problem Statement (July 13):** Framing the scope of the problem & structuring our approach & methodology to solving it.
3. **Ideating On Solutions (July 15):** Brainstorming possible techniques & algorithms to solve the problem. We had two similar problem statements with different approaches at this point in time.
4. **Data Collection & Type Exporting (July 17):** Understanding what data types best suit the many suitable algorithms & collating relevant datasets for the same.
5. **Algorithm Designing (July 18):** Finalizing on the algorithms to use & algorithms to experiment with to find out their possible involvement. We also finalized our approach to solving the problem.
6. **Prototyping Models (July 19):** Experimenting with the large set of chosen algorithms & datasets to see what works & with what. After much experimentation, we finalized on Logistic Regression, K-Nearest Neighbours, Support Vector Machine, Convolutional Neural Network & Random Forest Algorithm or Polynomial Regression to integrate the models with one another.
7. **Writing The Proposal (July 19) :** Created a preliminary outline on the entire scope of the project, detailing the goals and overview of the project. Offered an outlook on our potential tentative algorithms and accuracy models, along with detailed roles and other key project details.
8. **Testing Models (July 20):** Training & cross-validating the selected models developed post algorithm & subsystem prototyping to understand data flow & accuracies. Post testing of various models, we realized that the neither the Random Forest Algorithm nor Polynomial Regression were providing beneficial results due to the Random Forest's binary decision tree nature & the Polynomial's tendency to overfit, hence they were swapped out for a maximizing algorithm, similar to an Adaboosting algorithm.
9. **Implementing Final Model (July 26):** Applied a combined triple algorithmic approach using Logistic Regression, K-Nearest Neighbors, and Support Vector Machine on a split dataset to determine whether a plant was healthy or not. Coupled this with a Convolutional Neural Network for specific results on what disease the plant had using a single large dataset. We ran into some issues with requiring various datasets to train at first, but we were able to combat this after the addition of numpy array inputs & a better implementation of the one vs all method, as learnt in the feedback session.
10. **Finishing The Report, Video & Reflections (July 31):** Used the final model as a structure for the algorithmic implementations, which were improved through further testing. Created flowcharts & other relevant materials for submission, along with final predicted images for the results & conclusions. Added an image augmentation algorithm to improve training.

Algorithm Methodology (Methodology 2):



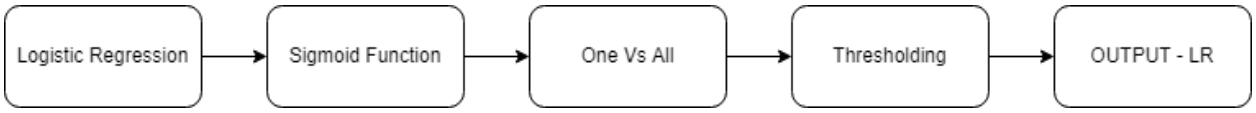
- **Why Logistic Regression?**
- Efficient Training & Number Of Observations > Number Of Features
- Used For Classification

- **Why K-Nearest Neighbours?**
- High Accuracy & Fast Multiclass Operations
- Used For Classification

- **Why Support Vector Machine?**
- Effective In High Dimensionality & Accounting For Errors
- Used For Classification

- **Why Convolutional Neural Network?**
- High Accuracy & Self-Adjusting Features
- Used For Parametrization & Classification

Logistic Regression

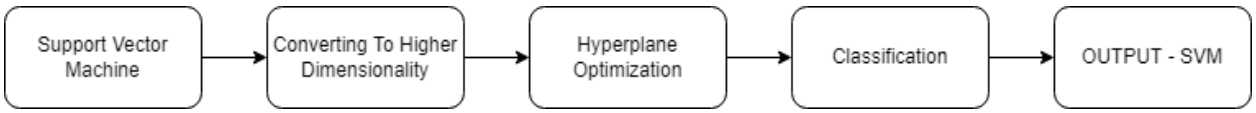


- **Sigmoid Function:**
- Plots An Shaped Curve To Generate A Probability Of The Data Being One Of Two Classes

- **One Vs All:**
- Continues The Sigmoid Function For Successive Classes One After The Other, Taking The Highest Probability

- **Thresholding**:
- Sets A Value Above Which The Result Is True For That Class & Below Which Which The Result Is False For That Class

Support Vector Machine



- **Converting To Higher Dimensionality:**
- Converting Data From 2D To 3D

- **Hyperplane Optimization**:
- Making A Plane With The Highest Euclidean Distance From Points In Order To Make Clusters

- **Classification**:
- Classifying Given Point With Available Clusters

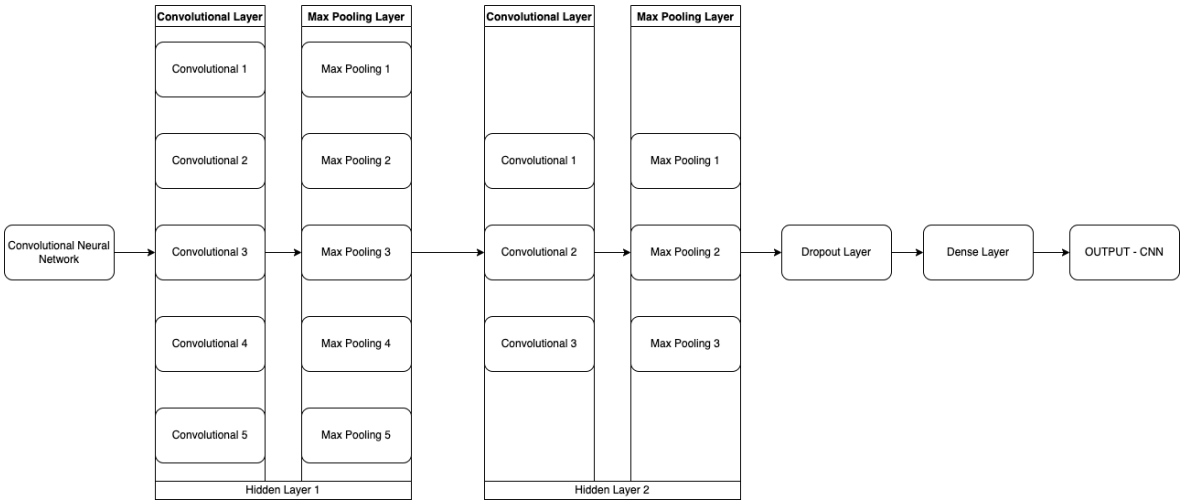
K- Nearest Neighbours

- **Euclidean Distancing:**
- Finds The Smallest Euclidean Distance Between Points

- **Finding The K Nearest Neighbours:**
- Denotes Points With Euclidean Distances Below A Certain Threshold As "Nearest Neighbours"

- **Clustering:**
- Groups "K" Nearest Neighbours & Classifies Given Point

Convolutional Neural Network

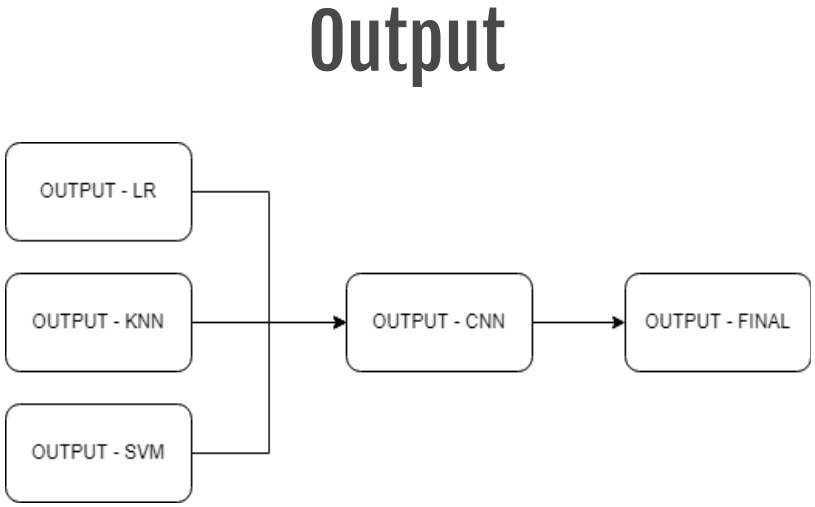


- **Convolutional Layer:**
- Learning features from the data, repeated for 5x5 & 3x3 matrix (standard)

- **Max Pooling Layer:**
- Gathering optimized features from each convolutional layer

- **Dropout Layer:**
- Removing outlier features from the data

- **Dense Layer:**
- Classifying the data based on features



- Broad classification group given by model with highest accuracy to CNN
- Images are augmented as well for training the CNN
- The output of the 3 models includes an F-1 score table & confusion matrix
- The final output includes the image & the classification

Division of Labor:

1. Adhish: Adhish's strength lies in his mathematical & computational thinking, hence he was responsible for the development & selection of the mathematical algorithms & final model, including but not limited to data flow, parameter setting & accuracy. He also worked on the code, with his prior experience in machine learning algorithms, he was able to quickly develop the algorithms. He worked closely with Tsitsi, who is in a similar timezone in order to simplify the workings of the model to algorithm flowcharts, with Johan to implement & execute computationally efficient & relevant code & with Jenny to position our project effectively amongst others & the science behind the project.
2. Johan: Johan's experience in hands-on software allowed him to understand the nitty gritties of memory efficiency debug efficiently. Hence, he was responsible for the quality assurance & fine execution of the required models, with his duties ranging from memory & speed tests to training the model for cross-examination. He worked closely with Jenny, who is in a similar timezone in order to understand the various data types we will be working with, with Adhish to implement relevant algorithms & and with Tsitsi to simplify the conversion of the workings of the model to paper & presentation.
3. Tsitsi: Tsitsi's ability to get the main idea across an ocean of complexity had found its calling in the documentation of our project in the report & video editing. Consequently, she used her understanding of audience perception to break down complex ideas & thoughts in the slides for the video as well as the proposal/report. She worked closely with Adhish, who is in a similar time zone, to create algorithm flowcharts, with Jenny to

represent the science behind the project & with Johan to simplify our explanation of the model.

4. Jenny: Jenny's forte lies in prioritization & getting into the deep details of concepts in order to make something big from something small. It is this resourcefulness that has shone through in the form of collecting, combining & interpreting datasets for our project. Simultaneously, she researched the biological aspects of our project coupled with competitive analyses of pre-existing products, being responsible for datasets & research. She worked closely with Johan, who is in a similar time zone, to manage the manipulation of data, with Adhish to optimize relevant subsystems in the model to stand out & with Tsitsi to demonstrate the underlying scientific principles.

Datasets:

Blemishes can often identify plant diseases on the leaves. We have chosen common illnesses that cause great ecological harm, have a high chance of going undetected & are widespread across the world. This is beneficial because many professions associated with plants already have certain systems in plant disease identification. Still, few people directly access plant disease identification in everyday life. The various existing models have not been trained with many significant diseases prevalent internationally, as opposed to being potentially biased from only national or limited international datasets. Therefore, in choosing these plants, we can allow more people to be more informed about the diseases in these plants and reduce potential ecological damage.

I. Dataset 1 [1709 Entries]
    A. Diseased [805 Entries]
    B. Healthy [904 Entries]
II. Dataset 2 [19,181 Entries]
    A. Apple [2526 Entries]
        1. Apple Scab [630 Entries]
        2. Black Rot [621 Entries]
        3. Cedar Apple Rust [275 Entries]
        4. Healthy [1000 Entries]
    B. Blueberry [1000 Entries]
        1. Healthy [1000 Entries]
    C. Cherry (Including Sour) [1854 Entries]
        1. Powdery Mildew [1000 Entries]
        2. Healthy [854 Entries]
    D. Corn (Maize) [3498 Entries]
        1. Cercospora Leaf Spot Gray Leaf Spot [513 Entries]
        2. Common Rust [1000 Entries]
        3. Northern Leaf Blight [985 Entries]
        4. Healthy [1000 Entries]
    E. Grape [3423 Entries]
        1. Black Rot [1000 Entries]
        2. Esca (Black Measles) [1000 Entries]
        3. Leaf Blight (Isariopsis Leaf Spot) [1000 Entries]
        4. Healthy [423 Entries]
    F. Orange [1000 Entries]

    1. Huanglongbing (Citrus Greening) [1000 Entries]
G. Peach [1360 Entries]
    1. Bacterial Spot [1000 Entries]
    2. Healthy [360 Entries]
H. Bell Pepper [1997 Entries]
    1. Bacterial Spot [997 Entries]
    2. Healthy [1000 Entries]
I. Potato [2152 Entries]
    1. Early Blight [1000 Entries]
    2. Late Blight [1000 Entries]
    3. Healthy [152 Entries]
J. Raspberry [371 Entries]
    1. Healthy [371 Entries]
K. Soybean [1000 Entries]
    1. Healthy [1000 Entries]
L. Squash [1000 Entries]
    1. Powdery Mildew [1000 Entries]
M. Strawberry [1456 Entries]
    1. Leaf Scorch [1000 Entries]
    2. Healthy [456 Entries]
N. Tomato [9325 Entries]
    1. Bacterial Spot [1000 Entries]
    2. Early Blight [1000 Entries]
    3. Late Blight [1000 Entries]
    4. Leaf Mold [952 Entries]
    5. Septoria Leaf Spot [1000 Entries]
    6. Spider Mites Two-Spotted Spider Mite [1000 Entries]
    7. Target Spot [1000 Entries]
    8. Yellow Leaf Curl Virus [1000 Entries]
    9. Mosaic Virus [373 Entries]
    10. Healthy [1000 Entries

Results:

Our results are split across two major sections for each of the Tiers and five sub-sections in total for all of them. They are outlined in detail below.

Tier 1 (LR, KNN, SVM)

Output Accuracy & Loss:

```
LR: 0.917341 (0.043566)
KNN: 0.930523 (0.025313)
SVM: 0.918076 (0.038037)
```

Results:

```
KNN has highest score. Using KNeighborsClassifier()
[1 0 0 1 1 0 0 1 0 0 1 1 0 1 1 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 1 1 0 0 1 0
 1 1 0 0 0 1 1 0 1 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 0
 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1 0 1 0 1
 0 1 1 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 0
 1 0 0 1 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 1 1 1
 0 0 0 0 1 0 1 1 1 1 1 1 0 0 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 0 0 1
 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 1 1 1 1 1 0 0 1 1 1 0 1 1 0 0 1
 0 1 0 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0
 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1
 0 1 1 0 1 1 0 1 0]
```

F-1 Scores:

	precision	recall	f1-score	support
0	0.95	0.89	0.92	157
1	0.91	0.96	0.93	185
accuracy			0.93	342
macro avg	0.93	0.92	0.93	342
weighted avg	0.93	0.93	0.93	342

```
Overall Accuracy score for Tier 1: 0.9269005847953217
```

Tier 2 (CNN)

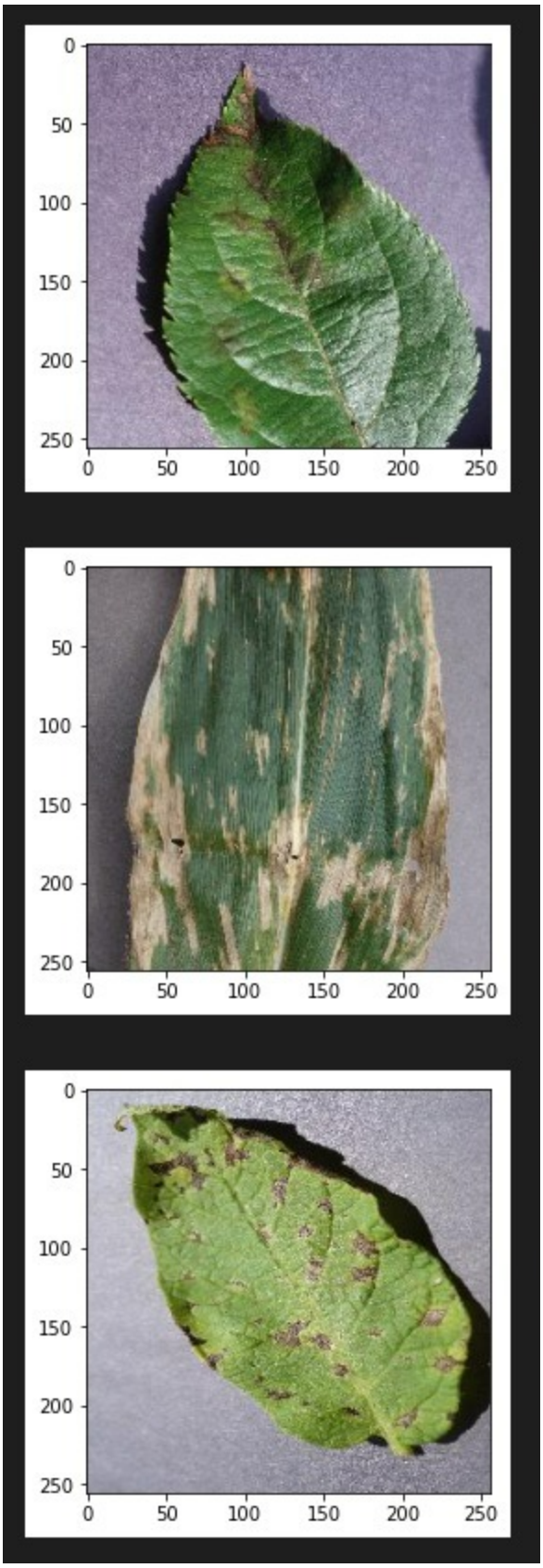
Output Accuracy & Loss:

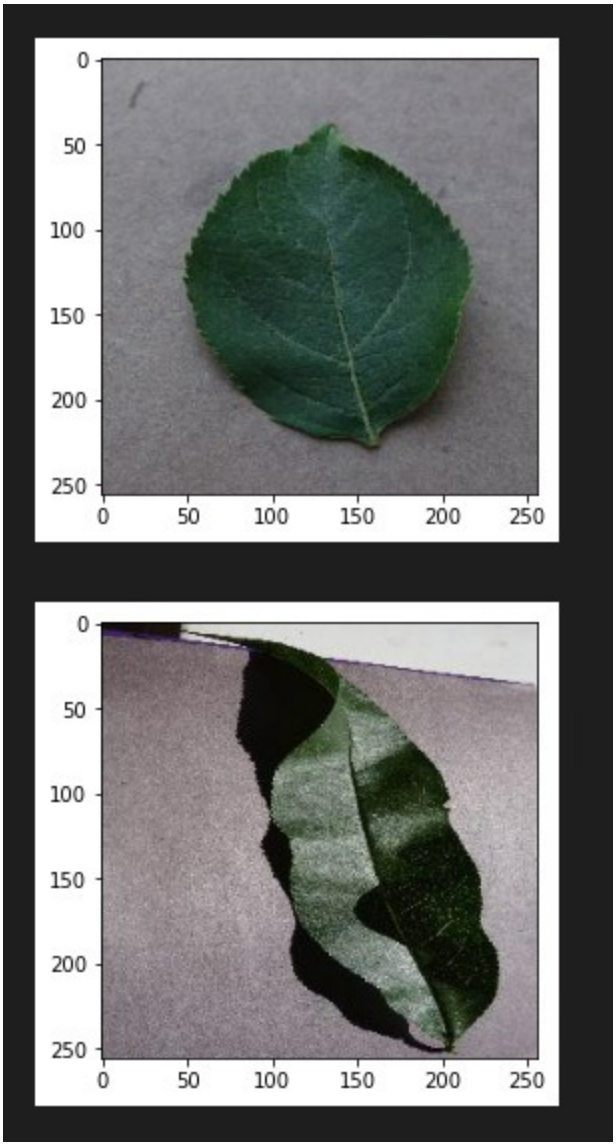
```
Epoch 1/2
946/946 [==============================] - ETA: 0s - loss: 1.1014 - accuracy: 0.6635WARNING:tensorflow:Learning
946/946 [==============================] - 544s 574ms/step - loss: 1.1014 - accuracy: 0.6635 - lr: 0.0010
Epoch 2/2
946/946 [==============================] - ETA: 0s - loss: 0.5037 - accuracy: 0.8378WARNING:tensorflow:Learning
946/946 [==============================] - 560s 591ms/step - loss: 0.5037 - accuracy: 0.8378 - lr: 0.0010
```

Results:

```
Apple_scab.jpg
1/1 [==============================] - 0s 44ms/step
Apple___Apple_scab
corn_leaf_spot.jpg
1/1 [==============================] - 0s 23ms/step
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot
early_potato.JPG
1/1 [==============================] - 0s 25ms/step
Potato___Early_blight
healthy_apple.JPG
1/1 [==============================] - 0s 26ms/step
Apple___healthy
healthy_peach.JPG
1/1 [==============================] - 0s 21ms/step
Peach___healthy
```

(All test images are correctly identified)


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

Results Analysis:

The results show accuracy as being 93% for the group, with precisions of 0.95 and 0.91, recall of 0.89 and 0.96, the F1-score as 0.92 and 0.93, and the support as being 157 and 185 for diseased vs. healthy, respectively. These results show that the code was extremely successful in overall algorithm accuracy and reliability. This feat can be attributed to our choice of algorithms and the actual implementation of these algorithms, as they are very well known for being successful. The accuracy scores for LR, KNN, and SVM were astounding at 91.7, 93.1, and 91.9, respectively, as seen in Tier 1. In our actual code, we ran each algorithm independently, but because we used the same dataset for LR, KNN, and SVM, their results were similar and great. For the CNN, it took the input image and determined if it was diseased and what disease it had, and based on the results we saw above, it was accurate. As a group, we had to modify the algorithm for a specific dataset for CNN, which led to better integration. In addition, we used TensorFlow metrics to determine the accuracy of our algorithm, which, as seen in the Tier 2 output accuracy & loss, is 0.66 and 0.84, respectively, for each of the Epochs. The low accuracy of the first score shows that more work needs to be done to further the results of our CNN algorithm. This may be partly because of the more in-depth CNN dataset, which differs from the significantly smaller dataset used for the other three algorithms. Despite this, the results are promising and show this algorithm has great potential. In fact, as mentioned above, the Tier 2 results were all correct for each of the five test images: Apple Scab, Corn Cercospora, Early Blight Potato, Healthy Apple, and Healthy Peach. To ensure quality, we had set a goal from the beginning to achieve an accuracy of at least 75%, with precision & recall values being greater than 0.75 as well. Fortunately, we were able to exceed this goal in the majority of our tests.

Conclusion:

By using the various combinations of algorithms outlined above, we contributed to a large-scale solution to detect various plant diseases. These types of programs are essential in agriculture, and this could potentially result in a decline in global food shortages and positively affect the economy by combating this devastation. We detailed and used concepts such as clustering, regression & classification throughout our work to implement our program. Reaching an accuracy of 82% was an achievement for the group, but we furthered this with our F-1 score, which is the average of precision and recall, which was 0.81.

Although these results are incredible, we faced various challenges along the way. Python notebook integrations proved to be a significant challenge, primarily due to all the dependencies our extensive collection of algorithms required. Lengthy runtimes only further added to this by hindering our progress. Combined with the vastly different time zones our group is located in, it was not easy to collaborate and work simultaneously. There were also several issues with different dataset types and keeping up with data flow while integrating all the models we used. Despite these challenges, our group persevered and produced our program through dedication and effort. We hope that our implementation will be beneficial for society and the world as a whole.

Potential Next Steps:

1. Hardware Implementation:
    a. Expands the scope of our project to include real-time data, facilitates a greater variety of features and error handling that could potentially be implemented, hence resulting in more higher accuracy & reliability
2. Implement Real-Time Training:
    a. Accounts for more & more random errors & broadens the scope of the project, hence increasing the overall accuracy & reliability after every use
3. Better Tuning Regularization:
    a. Reduces overfitting & hence improves the algorithm's accuracy
4. Self-Adjusting Adaboosting Weights:
    a. Accounts for random errors such as misinterpreted or unclear inputs & hence increases reliability
5. F-1 Score Boosting:
    a. Increases the reliability of the model