**How can machine learning be used in determining crop disease in plants?**

David Chen

Altrincham Grammar School for Boys, UK

*Abstract*— Plant diseases are defined as an abnormal growth or dysfunction within a plant. Crop diseases affect plants that are grown on a large scale can result in many different problems for society, ranging from a drastically lower economic output, to a lack of food supply in third world countries. Current plant disease identification is often done manually, which is time consuming and costly, and prone to error due to subjectiveness caused by humans. Machine learning has emerged as an effective approach for automated plant disease detection. The aim of this report is to examine how current machine learning approaches can be used for automatically detecting crop disease. We have applied different machine learning methods to a publically available dataset in order to detect crop disease (i.e. rust, leaf spot) in this case.

Keywords—Image processing, pattern recognition, machine learning, crop disease,

# Introduction

As the population of the globe increases exponentially, projected at nearly 10 billion by 2020[1], being able to balance converting unused land into habitable land, and farmable land becomes more difficult to manage. The rising problem of needing more farmland to feed the exponentially increasing population means that it is critical to maximise the gains from each farmland for improved food production. The current largest source of the loss of crops is crop disease. Crop plants are the largest sources for feeding the population. The use of crops ranges from eating crops like wheat in processed products, to feeding the animals that we will later be eating. Crop disease causes significant damages to the crops and reduces crop yield, leading to potential economic losses as well as a lack of food for the population. Therefore, it is becoming a global threat for food sustainability. To reduce this threat, there is a need for us to develop an effective way to identifying crop diseases as early as possible in order to increase food production.

A disease of a crop plant refers to an abnormal growth or dysfunction within a plant. When a plant is affected by diseases, it normally shows visual symptoms or detectable changes such as in colour, shape that occur on the leaves. As disease progression, the colours, size could change.

Currently, the most common technique used to identify crop disease is by manual visual identification. The problem with identification using the naked eye is that a lot of agronomists or surveyors are needed in order to cover a farm area effectively. This also brings in each person’s objective view. As one person may see crop disease in one plant, another may not. These inconsistencies combine resulting in different plants being wrongly flagged. In addition, sending out people to go onto the field in order to do manual surveying is very labour intensive and time consuming due to the sheer size of the fields. It is also very expensive to do this as they are often required to be trained regularly to keep update with new knowledge about diseases, as well as having to move them to different fields to do their work. This is particularly difficult to achieve, especially in rural countries, where it is hard for agronomists or surveyor to access the fields due to poor infrastructures such as roads, transport and funding issues.

To help alleviate this problem, the development of using image processing and machine learning in order to detect diseases in plants unravelled rapidly. Research was done on detecting Wheat Leaf rust using distinguished partial least squares and support vector machines (Wang, et al., 2016). Another approach was proposed, using random forests and hyperspectral imagery (Gao, Nuyttens, Lootens, He, & Pieters, 2018) to detect any weeds within a maize crop. While not the same as detecting crop disease itself, this method can be adapted to find the specific crop diseases by using the training sets for those. In itself, detecting weeds within a crop field is also very useful to farmers, as it identifies areas which are not productive and would not have any yield. This would allow the problem to be treated with quickly, to improve the entire crop field yield.

In this report, real publically available datasets for various crops and diseases will be used to train different machine learning algorithms to draw comparisons and to show how effective machine learning is in detecting crop disease.

The rest of this report is organised as follows. Section 2 presents the related work in automatic diagnosis of crop diseases using image processing and machine learning methods; Section 3 describes the machine learning methods used for determining crop disease in this report; Section 4 conducted experimental evaluation with real datasets; Section 5 concludes the work and highlights future work.

# Related work

There have been many efforts in studying the image processing techniques to identify and classify diseases like fungal disease symptoms on agriculture crops (Pujari, Yakkundimath, & Byadgi, 2015). For instance, a proposed method to detect and diagnose plant disease involved the use of convolutional neural networks (Ferentinos, 2018). Convolutional neural networks (CNN) is a type of neural network that is normally applied to analysing images. CNN itself requires very minimal pre-processing, which drastically reduces operation time. Other researchers utilised different methods, for example, using a kernel-based support vector machine (SVM) (Borra & Pukkela, 2018) to detect diseases like rust, tikka, powdery and downy mildew, late blight and early blight in groundnut, apple, potato and tomato plants. K-Nearest neighbours (KNN) was also utilised in different situations (Fuentes, Yoon, Kim, & Park, 2017) to detect tomato plant diseases, and pest recognition. Random forests were used in detection of grape diseases using pictures from a wild environment (Biswas, Avil, Sarangi, & Pappula, 2016). More experiments were carried out using colour and shape patterns in 425 images of 3 types of rice diseases, which was later separated into 4 distinct classes to further increase precision in detecting what type of disease is present. (Maharjan, Takahashi, & Zhang, 2011)

It would seem that using image classification to classify plant diseases should be perfect now. However the current models are not accurate due to low sample sizes and small representation of the diseases. There are also some limitations to using machine learning in order to detect crop disease at this moment. The first of which lies in the set of images used. These images may be trained to detect a specific type of disease, like rust. However this means that the classifier will only be trained to detect these diseases. For example if we were to give the classifier a set of images of the leaf spot when the classifier was trained on detecting rust, we would not receive an accurate prediction. This would then mean that another set of images would be needed to train for another specific disease, which would include thousands more images to run through the code. Extending on from this, multiple types of images are needed for diseases that spread throughout the plant. As some plants may have diseases that start from the stem before reaching the leaves, we would also need to include pictures of this. Therefore, more efforts should be made to obtain images from different perspectives. Another problem with training an accurate classifier is the picture quality used. Many different situations within a picture could potentially change the boundaries of the class of disease. Pictures can be taken from different angles, different lighting, in different seasons and in front of an environment background. As the classifier is trained based on the actual picture itself, it becomes harder to normalise what would be an ideal model for that specific disease.

There are also over 50000 classes of crop species and diseases. The practicality of training every single plant to every single disease is poor and a lot of time would be required to train every plant with each disease, even with the highest end computer specifications. With these many diseases it is also very likely several diseases have many similar features with each other, and the confidence of the classifier to correctly predict a disease would be lowered.

# The application of The machine learning methods to crop disease detection

The purpose of this report is to examine how existing commonly used machine learning approaches can be used to detect crop diseases and the level of their accuracy. In this section, we first introduce background of machine learning and seven selected machine learning methods used in this report such as KNN and random forests.

## Machine learning methods

You can give a general introduction about what is machine learning: for instance.

Machine learning is the field of study that gives the computer ability to learn without being explicitly programmed. Essentially, it is to build models that receives data and information from observations and real world interactions to learn the patterns from. It will then be able to provide accurate predictions based on the learned patterns of input data.

Broadly, machine learning can be divided into supervised learning and unsupervised learning. Supervised machine learning is what makes up most practical machine learning uses. Supervised learning utilises a function using a variable as an output dependent on the variable of the input. This is also commonly referred to

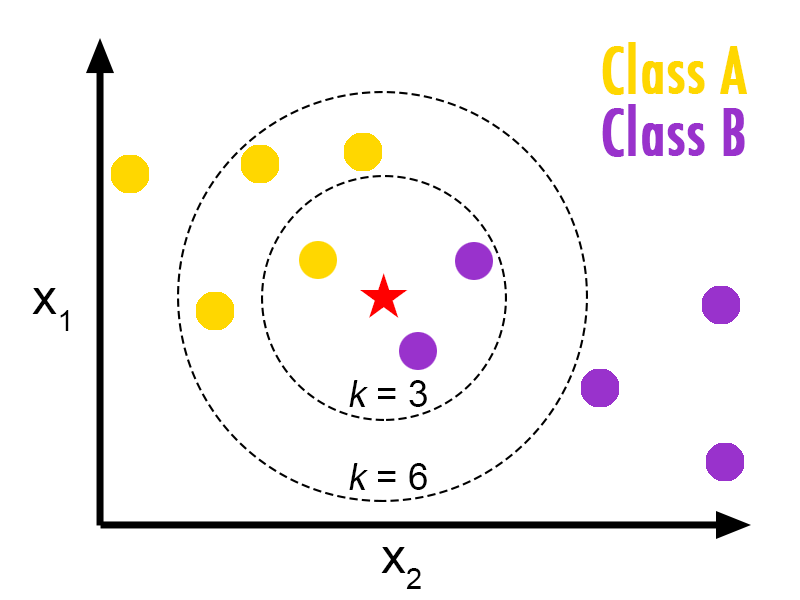
The goal is to approximate the function f, so that when new input data x is provided, the output data y can be predicted for that new input variable. It is supervised because the user know the correct answers y\* for the input x, and the algorithm makes predictions y on training data x, which the operator then corrects if it is wrong. The algorithm gradually learns from this until the predictions are acceptable. For example, the least of square error:

There are often two stages to consider when building supervised machine learning algorithms. These are the training, and testing stages. In the training stage, both input data and desired output are provided with feedbacks on the outcomes for building machine learning models. Once the training is completed, the trained models will be applied to the test data or new data during the testing phase. A popular example is the random forest algorithm.

In unsupervised learning, only the input data is provided without any output data. The point of this is to model the structure within the data to expose the patterns inside it. For unsupervised learning, it normally doesn’t need to provide desired output during the training stage. In this report, we only focus on supervised learning.

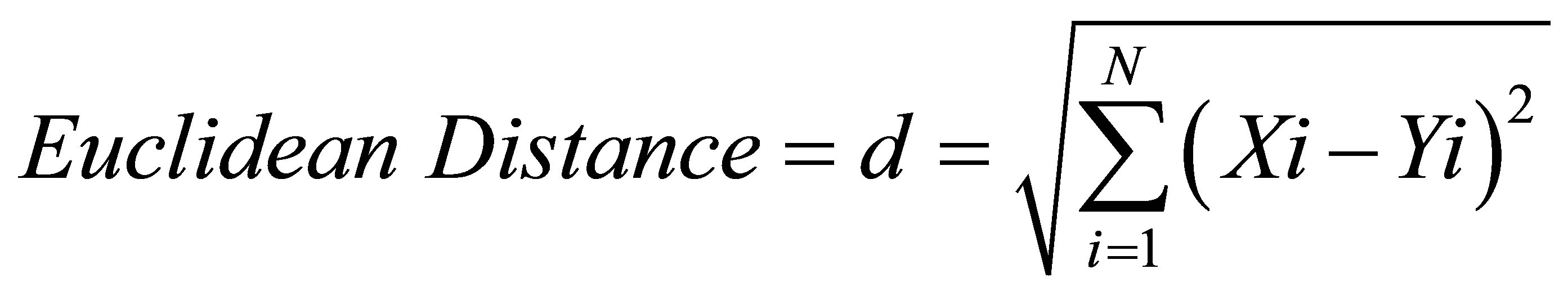
### KNN

KNN, or K-Nearest Neighbours [give the reference] is a popular supervised machine learning algorithm used commonly in pattern recognition. KNN utilises a graph that displays a pseudo distance between various points. Each training data point is assigned a class, which takes up a part of the graph. When a new point, or the test data, is inputted into the graph, we can set any number “k” of the nearest points. The number of each class of these nearest points, or neighbours, determines what the class of the new point is. This is visualised by the following:



***Figure 1. KNN algorithm concept [Reference 1] From: https://helloacm.com/a-short-introduction-to-k-nearest-neighbors-algorithm/***

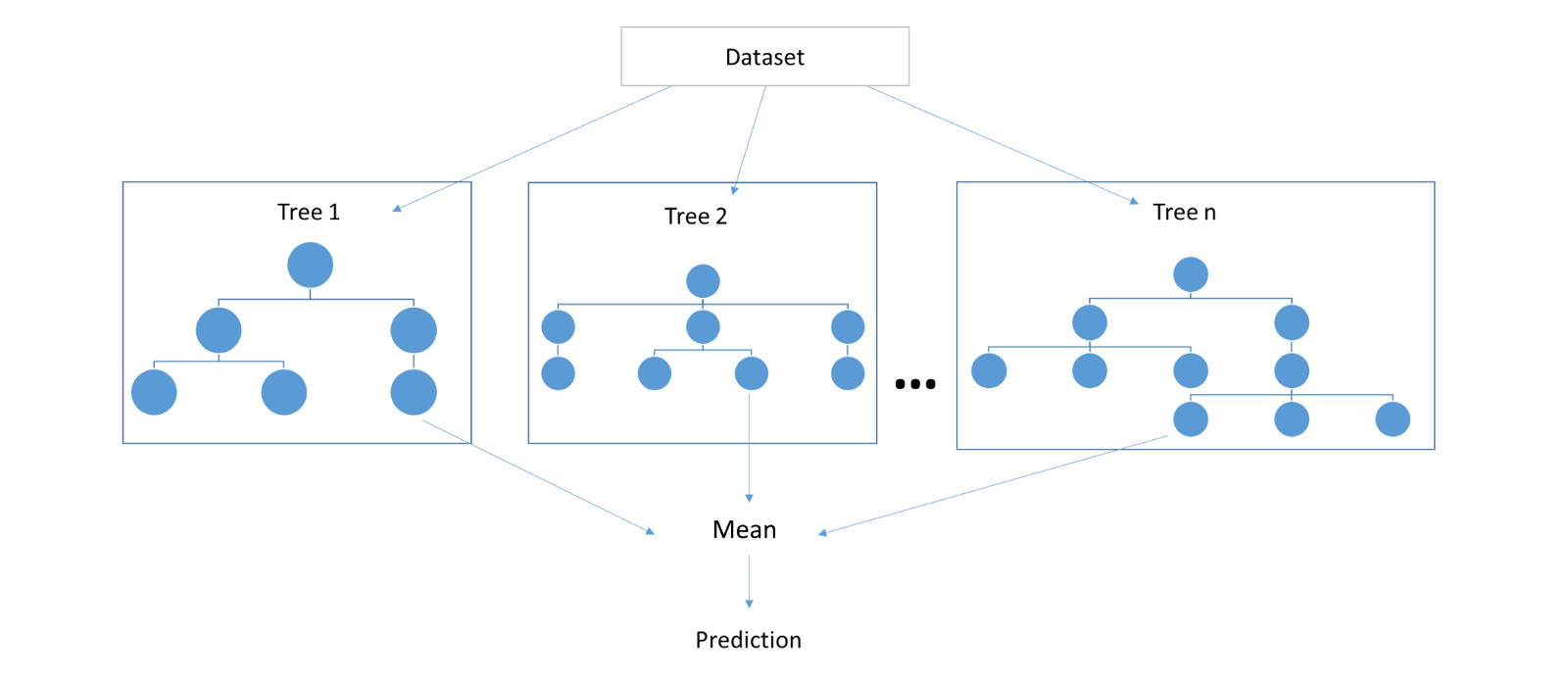
In this case, the pseudo distance between each point is also known as the Euclidean distance between each point as the data variables are continuous. The formula for the Euclidean distance is given as



Where we wish to find the difference between points X and Y in an nth dimension.

### Random Forest

Random Forest [give reference here too] is also another supervised machine learning method. It utilises the construction and merging of multiple decision trees during the training stage, which will all each output the class prediction of the test data during the testing stage. It is different from a normal decision tree on its own however, as instead of formulating a set of rules to make predictions, the random forest randomly selects observations in order to create several decision trees, and each result from each tree will be averaged.



***Figure 2 A random forest depiction, containing three separate decision trees [reference x]***

### Logistic Regression

Logistic regression [reference xhere] is a prediction method giving well calculated probabilities. In this context, there are multiple different classes, so a variant called Multinomial Logistic Regression will be used. Any useful equation orr visualised picture can be put here similar to the ones for KNN and random tree?

As this is a binary classifier, we need to train multiple classifiers in order to create an accurate model. When training the classifier for the disease Leaf Spot, we treat input data with this label as positive samples, and the other three classes as negative. Similarly, for the disease rust, we treat the data with this label as positive, whilst the other three are negative samples. This is done for all four categories during the training phase. During the prediction phase, the testing data is run against all four models to return their probability of whether it is that disease or not. Then, the highest probability class is chosen between the four different class predictions.

### Decision Tree classifier

A decision tree [references here] is a type of classifier that works by branching different nodes on a situation. Each internal node may represent a test on that situation. There three types of nodes that make up a decision tree, a decision node, chance node and end node. The decision tree breaks down a data sample into smaller subsets, to create an end product of decision and leaf nodes(node that doesn’t split). Each decision node has two or more branches. Leaf nodes represent a classification. The highest node in the tree is also named as the root node. After splitting the data, the tree is pruned so that the size of tree is reduced. This is often done when the decisions after a node is more often than not one choice, therefore that branch is usually replaced by that one choice.

### Linear discriminant analysis (LDA)

The LDA [reference here] is a technique that is a preferred method when there are more than two classes, such as with the case in this report. Linear discriminant analysis (LDA) assumes that each piece of data has a normal/Gaussian distribution. Gaussian data plots a bell like curve on a graph. LDA also assumes each attribute having the same variance, and each value vary around the mean by the same on average. Basing on these assumptions, LDA estimates mean and variance for each class. LDA converts a dataset in an n-dimensional space into a smaller subspace. The more dimensional space means the more accurate we can be in determining which point belongs to which class. LDA uses the information from the nth amount of axis to create a new axis, and the data is now projected onto this axis to maximize separation between the 4 categories. The new axis is after finding the central point of all data, the distance between each central point of each category and dividing the sum of the distances to the central point squared by the sum of all the scatter for each category.

Figure 3 An example chart utilizing LDA showing how a new axis would be drawn for two categories in 2 dimensions

### GaussianNB(Naïve Bayes)

Naïve Bayes[reference here] is a collection of algorithms that classifies all features independently. It uses the probability of each independent feature appearing in a data sample by matching with the training set. For example, a data sample with the categories shape and colour, matching to either an apple or banana, would predict a new piece of data as an apple if it was coloured red and had a round shape. This is due to the probability of an apple being red vs a banana being red. Similarly in crop disease, we can find the probability that a disease is rust if the shape of the outline of the colour contrast is a dot.

### Support vector machine (SVM)

SVM’s objective is to find a hyperplane(space within a dimension that has one less dimension than the dimension it is set in) in a set dimension of space, where the dimension is the same as the number of features, which distinctly classifies the data points. When dealing with multiple classes in a dataset, SVM often uses the pairwise/one-vs-one feature. In this one-vs-one feature, the idea is to develop a number of binary classifiers while being trained only on two classes. The formula to determine how many classifiers will be trained is given by

https://i.gyazo.com/fb45748b99fc1426ea8a98e57d8665ad.png

Where n is equal to the number of classes. For example, if there were four classes, there would be (4\*3)/2 = 6 classifiers. They would pair up A and B, A and C, A and D etc. If a new input test variable Y was inputted, and the class was unknown, Y would be presented to each classifier. The highest confidence probability will be taken as the output.

## The proposed framwork for crop disease detection

The diagram bellow shows the outline of the training and testing stages.



Starting with the input image, the training sample is inputted as a whole to the model.

It will then be processed in image processing by firstly splitting the data into the testing and training samples, before generating a set of features from the test data using various functions such as Haralick Texture and Hu moments. These features would then be labelled to the class it was present in. The classifier would learn how each set of features corresponds to each label, effectively generating the feature class for each disease. The features would then be normalised in order to have a standard distribution and suppress the large variance between the lowest weighted feature and the highest weighted feature.

After the classifier has been trained on the training images, it creates a model with set feature parameters for each disease label. This model is now test ready.

For testing, the test images from before are inputted. It is important that this set of data were never trained on the classifier as trying to predict something that was used as training material would be pointless. The classifier here extracts the features from each image and applies the values for these features to the disease parameters set by the model earlier. The highest match in these features will be selected as the final disease for which the model has predicted.

The following diagram outlines the stages that results in a class prediction in more detail.

Trained scaler

Labelling and Random Sampling

Using Train\_test\_split to split the image files into training and testing data

Labelled train image samples

Organised image files

Labelled test image samples

Feature generation

Train image features

Train feature normalization

Feature generation

Test image features

Test feature normalization

Normalized train features

Kfold cross validation

Normalized test features

Test data classify

Classifier training

Scores

Classify results esults:scores

Trained classifier

Sampling.PY

Train Sampling Result.h5

Test Sampling\_result.h5

Image\_feature\_generation.PY

Train\_features.h5

*Traned\_scaler.h5*

Train\_RF.PY

*Traned\_classifier.h5*

Cross\_validation\_test.PY

Test\_RF.PY

Figure 4.

# The Implementation and experimental evaluation

## The implementation

In preparation for the experimental process, I first installed a free and open source distribution for Python, named Anaconda. Within Anaconda are various options that may allow me to easily install and apply different modules that I will need to solve this case. Within Anaconda are several environments for Python for which I can set up different situations. There will be several different programs that will be created in order for this experiment to function. A couple of these will be used during the training phase. During the testing phase there will be separate Python applications for each separate machine learning algorithm that will be analysed.

## The experimental evaluation

### Datasets

There are 5 different plants that will be investigated, each with its own number of diseases. There are many samples of each disease for each plant.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Apple** | Apple scab | Cedar apple rust | Frogeye spot | Healthy |
| Number of samples | 630 | 275 | 621 | 1645 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Corn** | Leaf spot | Northern leaf blight | Rust | Healthy |
| Number of samples | 513 | 985 | 1192 | 1162 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Grape** | Black rot | Esca black measles | Leaf blight/ isariopsis leaf spot | Healthy |
| Number of samples | 1180 | 1383 | 1076 | 423 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Potato** | Early blight | Late blight | Healthy |
| Number of samples | 1000 | 1000 | 152 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tomato** | Bacterial spot | Early blight | Late blight | Leaf mold | Septoria leaf spot | Spotted spider mite | Target spot | Mosaic virus | Yellow leaf curl | Healthy |
| Number of samples | 2127 | 1000 | 1909 | 952 | 1771 | 1676 | 1404 | 373 | 5357 | 1691 |

Each classifier will be trained onto each of these diseases.

### Evaluation metrics

In order to measure accuracy, the accuracy measure is based on the standard metrics as follows:

accuracy = (tp+tn)/(tp+fp+tn+fn)

Where “*tp”* means true positive, “*tn”* represents true negative, “*fp”* represents false positive and “*fn”* represents false negative.

* A true positive is a positive sample that was predicted as positive by the classifier. (E.g. if a disease was rust, and the classifier predicted it as rust)
* A true negative is a negative sample being predicted as negative by the classifier. (E.g. if a disease was not rust and the classifier predicted as not rust)
* A false positive is a negative example being classified as positive by the classifier. (E.g. if disease was not rust, but the classifier predicted it as rust)
* A false negative is a positive example being classified as negative by the classifier. (E.g. if the disease was rust but the classifier predicted it as not rust)

### Experimental setup

The experiment was run on a computer with the following specs:

Processor: Intel(R) Core(TM) i3-4130 CPU @ 3.40GHz, 3400 Mhz, 2 Core(s), 4 Logical Processor(s)  
Operating system: Windows 10 Home  
Random access memory(RAM):8GB  
Graphics card: Nvidia Geforce GTX 1060

The following programs will be run:

* 1. featuregen.py: this program pre-defines the feature generation method and data saving method. This program then will be imported into the training and testing programs.
  2. Sampling6.py: this program will firstly read all the sample file names. It will then input the dataset folder (here is: dataset/Corn) to a list and label each sample numerically while saving the labels to another list. By using the train\_test\_split function, the train and test sample files with names and labels will be generated. This program will then save those data lists in two data files (“output/Corn\_train\_samples6.h5”, and “output/Corn\_test\_samples6.h5). The test sample set size is 0.2(20%) of all total samples.
  3. image\_feature\_generation6.py: this program reads the train samples created from the sampling result file (“output/Corn\_train\_samples6.h5”) and generates the global training features. The features are saved in an h5f file to be imported later. (“output/Corn\_train\_features6.h5”).
  4. method\_compare6.py: this program uses a 10 fold cross validation method (kfold and cross\_val\_score function) to compare several machine learning methods in disease detection on the training features file generated in section 2.3. The features are normalised with respect to 0-1 before undergoing validation.
  5. Train\_RF61.py: this program uses the training sample features dataset generated in section 2.3 to train the scaler and various classifier models. The classifier is trained by the normalised global training features which is read from the h5f file generated earlier. The trained scaler and classifier are saved in two files: “output/Corn\_train\_scaler.h5” and “output/Corn\_train\_model.h5”, respectively.
  6. Test\_RF61.py: this program tests one machine learning method, random forests in this case, in detecting disease using the test data features. The global test features data is generated from the samples specified in the list in test sample file (“output/Corn\_test\_samples6.h5) using the self predefined function “feature\_generate” in section 1. Generated global test features are normalized by the scaler which is trained by train globe feature data in section 2.5(“output/Corn\_train\_scaler.h5”). The results are used to calculate the real positive, real negative, false positive and false negative results.

The files are named “RF” to begin with due to the fact that the machine learning approach random forests will be the first one to be tested. To facilitate new versions of the program, the sample data files are organised as such:

1. Using the class name as the subfolder name of the sample files of this class
2. Using the plant name as the plant/crops data folder
3. The folder must be under the same folder of the code written

This is done so for convenience sake. Using this modification, any plant/crop disease can be tested on without modifying the actual code. The only requirement will be to input the name of the plant/crop when the program is running.

The programs will be run in the following order.

1. Sampling6.py to split samples into training and testing datasets
2. Image\_feature\_generation6.py to generate the training sample features
3. Image\_feature\_generation6.py to generate testing sample features
4. Method\_compare6.py to compare the 7 machine learning methods using k-fold cross validation
5. Train\_RF61.py to train the random forest classifier and scaler(can be switched to any other ML algorithm)
6. Test\_RF62.py to test the trained classifier and scaler using the testing sample dataset.

### Experimental results

For the 5 different plants, 5 data files were generated for each plant, saved in the corresponding subfolder of the plant.

Train\_samples6.h5 : training sample data file;

Test\_samples6.h5 : testing sample data file;

Train\_features6.h5: training features data file;

Test\_features6.h5 : testing features data file;

Train\_model: trained classifier file;

Train\_scaler: trained scaler file.

Running results ( test size=0.2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Corn** |  |  |  |
|  | Accuracy | STD | Min Accuracy | Max Accuracy |
| LR | 0.925674 | 0.011656 | 0.914018 | 0.93733 |
| LDA | 0.913341 | 0.016099 | 0.897242 | 0.92944 |
| KNN | 0.917234 | 0.011086 | 0.906148 | 0.92832 |
| CART | 0.920472 | 0.016212 | 0.90426 | 0.936684 |
| RF | 0.953582 | 0.010389 | 0.943193 | 0.963971 |
| NB | 0.893534 | 0.014619 | 0.878915 | 0.908153 |
| SVM | 0.846477 | 0.020133 | 0.826344 | 0.86661 |
|  |  |  |  |  |
|  | **Potato** |  |  |  |
|  | Accuracy | STD | Min Accuracy | Max Accuracy |
| LR | 0.972695 | 0.013259 | 0.959436 | 0.985954 |
| LDA | 0.957004 | 0.017475 | 0.939529 | 0.974479 |
| KNN | 0.976761 | 0.012188 | 0.964573 | 0.988949 |
| CART | 0.957585 | 0.01582 | 0.941765 | 0.973405 |
| RF | 0.981405 | 0.010004 | 0.971401 | 0.991409 |
| NB | 0.879698 | 0.018676 | 0.861022 | 0.898374 |
| SVM | 0.883193 | 0.027125 | 0.856068 | 0.910318 |
|  |  |  |  |  |
|  | **Tomato** |  |  |  |
|  | Accuracy | STD | Min Accuracy | Max Accuracy |
| LR | 0.898747 | 0.006125 | 0.892622 | 0.904872 |
| LDA | 0.880643 | 0.008943 | 0.8717 | 0.889586 |
| KNN | 0.936467 | 0.005799 | 0.930668 | 0.942266 |
| CART | 0.859788 | 0.008975 | 0.850813 | 0.868763 |
| RF | 0.971572 | 0.003109 | 0.968463 | 0.974681 |
| NB | 0.628512 | 0.017224 | 0.611288 | 0.645736 |
| SVM | 0.663616 | 0.009244 | 0.654372 | 0.67286 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Apple** |  |  |  |
|  | Accuracy | STD | Min Accuracy | Max Accuracy |
| LR | 0.958204 | 0.012988 | 0.945216 | 0.971192 |
| LDA | 0.950316 | 0.009198 | 0.941118 | 0.959514 |
| KNN | 0.959001 | 0.012945 | 0.946056 | 0.971946 |
| CART | 0.912857 | 0.012997 | 0.89986 | 0.925854 |
| RF | 0.981071 | 0.004932 | 0.976139 | 0.986003 |
| NB | 0.858833 | 0.021734 | 0.837099 | 0.880567 |
| SVM | 0.633315 | 0.032725 | 0.60059 | 0.66604 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Grape** |  |  |  |
|  | Accuracy | STD | Min Accuracy | Max Accuracy |
| LR | 0.949522 | 0.004612 | 0.94491 | 0.954134 |
| LDA | 0.952911 | 0.011673 | 0.941238 | 0.964584 |
| KNN | 0.959682 | 0.007067 | 0.952615 | 0.966749 |
| CART | 0.912897 | 0.020033 | 0.892864 | 0.93293 |
| RF | 0.976609 | 0.006906 | 0.969703 | 0.983515 |
| NB | 0.870433 | 0.025534 | 0.844899 | 0.895967 |
| SVM | 0.701443 | 0.028775 | 0.672668 | 0.730218 |

# Conclusion

In conclusion, with respect to the experimental results, we can find that the random forests algorithm provides the very best accuracy for every single plant and its diseases. KNN follows random forests as the second highest accuracy, however it struggles with corn diseases. Decision Tree classifier is bad when classifying 9 classes, as seen in the tomato investigation. This suggests that CART is poor when dealing with multiclass classifications. GaussianNB and Support Vector Machines provide poor accuracy, hinting that these algorithms are not ideal when classifying crop diseases. SVM also requires a lot more computational power and time in comparison to the other methods, when increasing sample size. Therefore we can conclude that random forests and KNN would be the top methods for classifying crop disease.

# References

Biswas, S., Avil, S., Sarangi, S., & Pappula, S. (2016). *Random forest based classification of diseases in grapes from images captured in uncontrolled environments.* IEEE.

Borra, S., & Pukkela, P. (2018). *Machine Learning Based Plant Leaf Disease Detection and Severity Assessment Techniques: State-of-the-Art.* ResearchGate.

Fang, Y., & Ramasamy, R. (2015). *Current and Prospective Methods for Plant Disease Detection.* Athens: University of Georgia.

Ferentinos, K. (2018). *Deep learning models for plant disease detection and diagnosis.* Elsevier.

Fuentes, A., Yoon, S., Kim, S. C., & Park, D. S. (2017). *A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests RecognitionA Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition.* US national library of Medicine .

Gao, J., Nuyttens, D., Lootens, P., He, Y., & Pieters, J. (2018). *Recognising weeds in a maize crop using a random forest machine learning algorithms and near infrared snapshot mosaic hyperspectral imagery.* Elsevier.

Maharjan, G., Takahashi, T., & Zhang, S. (2011). *Classification Methods Based on Pattern Discrimination models for web-based diagnosis of rice diseases.* David Publishing.

Martinelli, F., Scalenghe, R., Davino, S., Panno, S., Scuderi, G., Ruisi, P., . . . Dandekar, A. (2014). *Advanced methods of plant disease detection.* France: INRA and Springer-Verlag .

Patil, S., Chandavale, A., & Dixit, B. (2015). *Recognition of plant disease.* Pune: MIT College of Engineering.

Pujari, J., Yakkundimath, R., & Byadgi, A. (2015). *Image Processing Based Detection of Fungal Diseases in Plants.* Elsevier.

Wang, H., Qin, F., Ruan, L., Wang, R., Liu, Q., Ma, Z., . . . Wang, H. (2016). *Identification and Severity Determination of Wheat Stripe Rust and Wheat Leaf Rust Based on Hyperspectral Data Acquired Using a Black-Paper-Based Measuring Method.* PLOS.

It’s important to recognise crop disease as one outbreak could lead to many people either emigrating away from a country, or dying from starvation. This is hugely harmful to any country. An example would be the disease named “late blight”. This disease is responsible for destroying more than half of the tomato crop in the United States. It is also famous for causing the Great Famine in Ireland in the mid-19th century. This can also happen for many other diseases, as crop diseases are not limited to blight. Pathogens consisting of much fungi, virus, and bacteria can all infect any parts of a plant. These infections and diseases may also be highly contagious, which can devastate an entire crop field incredibly fast. The use of pattern recognition itself when examining the crops are currently being looked at in a number of different settings, like identifying weeds, and discolouration in citrus fruit. There are many viable methods to detect crop disease, and therefore the aim of this project is to highlight the efficiency of Machine Learning, and specifically, pattern recognition, in this application.

# Current research

The research on the concept of using image processing for detecting crop disease has started relatively recently( “Digital image processing techniques for detecting, quantifying and classifying plant diseases”, Jayme Garcia Arnal Barbedo, 2013). When it comes to mobile devices, the research began even more recently. This could be due to the inferior computing capacity that you can have on a mobile cellular device, compared to a desktop computer. However, as the processing power of mobile devices become stronger, the ability to perform complex image processing becomes easier to achieve. Alternatively, a different method that could be used in the future includes having a cloud server for which image taken with cellular devices can send the pictures to. This cloud server would contain a mainframe computer that would do the processing before sending the results back to the user. This alternative method has problems that stem from being able to stay in range of a wireless network, which would require a long time to set up, and is also expensive.

Sagar Patil, Bharati Dixit and Anjali Chandavale (Patil, Chandavale, & Dixit, 2015) conducted a research into automatic detection of plant diseases with which their intensity of colours is the same, however the colours themselves are different. This method takes into account the RGB values of each pixel in the picture to classify whether each picture contains a disease or not. The median filter technique was applied to smoothen the image, and the Otsu thresholding method is used for calculating boundaries between different diseases.

Yi Fang and Ramaraja Ramasamy (Fang & Ramasamy, 2015) listed many ways to detect plant disease, most of which included biological tests such as the Enzyme Linked Immunosorbent Assay or ELISA test. These tests are very accurate, as the antigens from the virus, bacteria or fungi can only bind to one shape of antibody. This assay contains the specific shaped antibody, meaning this method is almost always correct when it comes to determining whether a plant is a carrier of the pathogen which can cause a specific disease. A couple methods that are less manual and rely on imaging, include methods such as thermography. This method takes into account the difference in colour when an infrared picture of a plant is taken via thermographic cameras. The resulting disease is carefully monitored using thermographic imaging, and the amount of water lost is able to be calculated, without changing the surrounding environment.

Another paper (Martinelli, et al., 2014) reviews several methods, based around nucleic acid-based. As some pathogen detection methods are based upon DNA, such as fluorescence in situ hybridization (FISH) and many PCR variants. Others are RNA based, such as reverse transcriptase-PCR, and nucleic acid sequence-based amplification (NASBA). These methods overcome uncertain diagnosis, enabling a fast and accurate detection and quantification of pathogens. This method however is only efficient when running diagnosis one a few specimens. When taking into the account of large areas to test for pathogens, it is not the most efficient method to go around. They also review different methods such as lateral flow microarrays (LFM). These allow fast hybridization based nucleic acid detection by using a colorimetric signal. The arrays are built on lateral flow chromatography nitrocellulose membranes, and have detection limits similar to microarrays, and are able to reduce the need of expensive equipment. However this technology is not widely available, and so not many people are able to utilise this method.

# Proposed method

The following diagram shows the flowchart of the training process, which we will use in order to train the classifier

Testing stage

Classification

Feature extraction

During the input stage, training images that contain pictures of plants with crop disease and without crop disease are inputted in the program.

PSEUDOCODE:

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

DATASET= “dataset/trainingset”  
 Pictures=[]

For picture in DATASET:

Pictures.append(picture)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~