Vrije Universiteit Amsterdam

X 400213

Project Optimization of Business Processes

Deep Learning for Bacterial Spot Prediction

Author:
Apala M Saha
Kostantinos Sakellariou

Supervisor:
Yura Perugachi-Diaz

Contents

1 Introduction

There has been extensive studies and research to protect plants from diseases in the field of agriculture. Understanding the health of the plants are extremely important for the economy of the agricultural field as bacterial spots is a common disease in plants which plays as a major limiting factor in agriculture. Protecting the plants from bacterial spots is crucial for improving the quantity and the quality of the plants. Bacterial spots from one plant can easily contaminate the nearby plants if not treated properly at an early stage. This affects the production of crops and the economy at a broader spectrum. Thus, providing early detection and identification of the bacterial spots is helpful in choosing the treating for the plants at an early stage and stopping the bacterial spots to spread to other plants. However, analyzing each plant leaf for disease and pest can be extremely time-consuming. Thus, during this project, we try to build a DSS for the users in the agriculture for them to better understand the leaves with bacterial spots so that quick action can be taken to treat the already infected plants. Thus, the main objective of this project is to detect the bacterial spots at an early stage in order to provide the appropriate treatments on time.

2 Project goal and objective

In this project, we have used machine learning and computer vision to build an image classifier which will help us classify the images with bacterial spots from the other images present in the data set. Deep learning is a new trend in machine learning and it helps computer models learn to perform classification tasks from images. It helps in achieving good accuracy. Here, models are trained with the help of a large amount of labelled data set and neural network architectures using multiple layers. Our goal is to build a Decision Support System for users in the field of agriculture. By creating deep learning models we will help people differentiate between the healthy leaves and the leaves containing bacterial spots on them. So our prediction will help the owners of the plants to identify the infected plants correctly and take precautionary measures accordingly. This will allow the users to make faster and more informed decisions as well.

3 Background information available

We have been provided with the data-set of around 16K images that we require to use for image classification. This data-set has images of 256 x 256 pixels and the images have been divided into 15 different sub-classes with three main classes i.e., tomato, bell pepper and potato.

4 Scientific Explanation of Algorithms

In the following subsections, the algorithms/methods used for the image classification are discussed extensively:

Before we start modelling, we need to analyze the database containing three different classes of plants distinctly separated in a set of 15 different sub-classes. We resize the images of these plants into 32 X 32 x 3 RGB images and perform the model predictions and optimization on these downscaled images.

We divide the whole data set into two parts(70% of the data set is used for training while 30% of the data set is used as a test set)

To resize the data provided to us we used the following functions:

1. os.path.join
2. resize
3. os.path.splitext

For classifying the various images of the leaves as healthy and unhealthy, we followed two different types of modeling approaches:

1. Convolutional neural network
2. Logistic Regression

Explaining in brief about the models used in the project:

4.1 Logistic regression

Logistic Regression is one of the simple and commonly used machine learning models which can be used for classification problems. It is easy to implement. Here we have used the python libraries scikit-learn to complete the modelling for the image classification. We tried to create the logistic regression model using Tensorflow and the library known as Keras. However, we were coming across some unforeseeable errors regarding that due to which we went forwarded and created the model with python libraries.

In this code, we separated the data-set and the categories and provided them as inputs in the form of lists. The list is later converted into the form of an array and each image is put under the specified subclass that it originally belonged to. As logistic regression is unable to take words in the form of input, we change the name of the sub-classes by initializing them in the form of an integer value ranging from 0 to 14. Then the data-set is divided into a train set and test set for the model by using the train_test_split function which randomly splits the data between two parts i.e., 70% as train data-set and 30% as test data-set. The model is then trained with the training data-set and we get an accuracy of 66% We also need to make a few more settings before we train the model. The following steps need to be taken into consideration:

1. Loss function— It will help us understand how accurate the model is during the training. categorical_cross entropy is used here.
2. Optimizer — This tells us how the model will be updated based on the data and the loss functions. We have used the default solver as the optimizer
3. Metrics — Helps to monitor the training and testing steps.

To give a brief overview of how our model worked, Fig(1) shows the accuracy that the model provided:

	precision	recall	f1-score	support
0	0.57	0.36	0.44	250
1	0.79	0.79	0.79	348
2	0.82	0.85	0.84	224
3	0.45	0.12	0.19	41
4	0.53	0.57	0.55	251
5	0.65	0.72	0.68	510
6	0.43	0.23	0.30	260
7	0.71	0.73	0.72	374
8	0.60	0.62	0.61	481
9	0.59	0.65	0.62	232
10	0.57	0.54	0.56	439
11	0.52	0.57	0.55	312
12	0.76	0.54	0.63	110
13	0.67	0.74	0.70	397
14	0.79	0.89	0.84	726
accuracy			0.66	4955
macro avg	0.63	0.60	0.60	4955
weighted avg	0.65	0.66	0.65	4955

Figure 1: Accuracy of the model created

We have created a confusion matrix based on the model that we have trained. The confusion matrix helps in providing a tabular format to visualise the performance of the algorithm in supervised learning. The Fig(2) explains the confusion matrix that we got from the logistic regression:

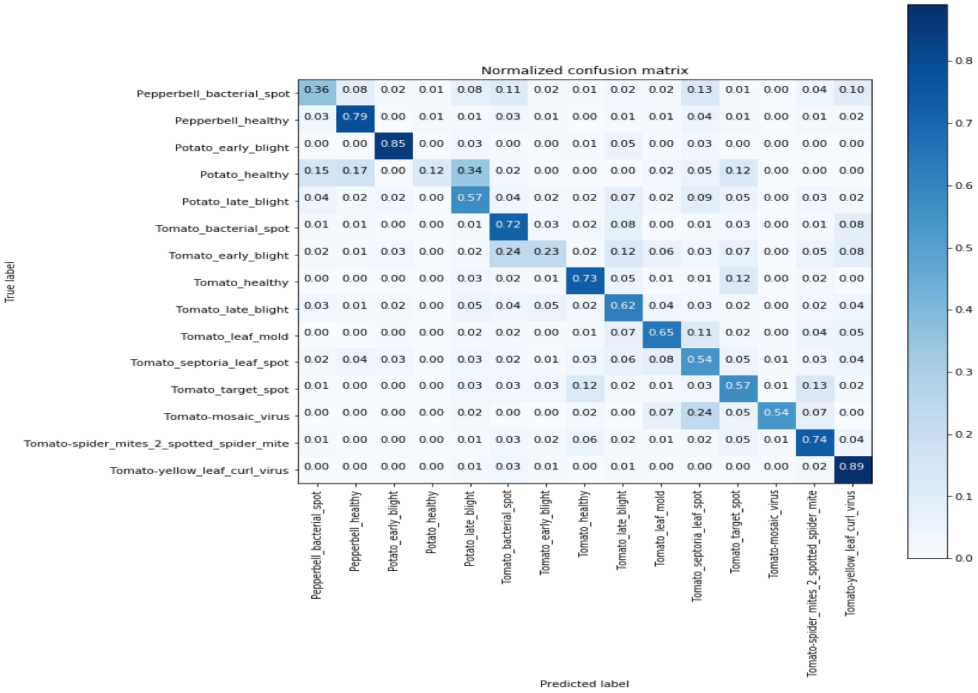


Figure 2: Confusion matrix of the logistic regression

4.2 Convolutional Neural Network

As explained in the papers (Deep Learning for Tomato Diseases: Classification and Symptoms Visualization, Mohammad Brahmni, 2017 and A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition, Alvaro Fuentes), it has been explained that CNN is a powerful model to be used for image classification on large data sets. Thus, following the same reference, we use the CNN model for the image classification of the leaves with bacterial spots. As the convolutional neural network is a deep neural network that works impressively with image classification, we create the image classifier using the TensorFlow by implementing the CNN. CNN is constructed with the help of multiple convolution layers and pooling layers. Our convolutional neural network is built with the following architecture as shown in Fig(3):

Name	**Kernel**	**Stride**	**Padding**	**Channels In/Out**
conv1	3×3	1	1	3/32
maxpool1	3×3	2	1	32/32
conv2	3×3	1	1	32/64
maxpool2	3×3	2	1	64/64
conv3	3×3	1	1	64/128
maxpool3	3×3	2	1	128/128
conv4_a	3×3	1	1	128/256
conv4_b	3×3	1	1	256/256
maxpool4	3×3	2	1	256/256
conv5_a	3×3	1	1	256/512
conv5_b	3×3	1	1	512/512
maxpool5	3×3	2	1	512/512
avgpool	1×1	1	0	512/512
linear	–	–	–	512/15

Figure 3: Architecture used for CNN

The convolution layer helps in transforming the input image to extract the features and distinguish them correctly. A kernel is used to convolve the images as it is specialised to extract features. We can also apply multiple kernels to a single image for understanding the multiple features.ImagedataGenerator is used for data augmentation for the images of the similar leaves taken in different angles. This helps the network to learn more robust features. An activation function is applied to the convoluted values which help in increasing the non-linearity. The activation function used for this model is ReLU layer.

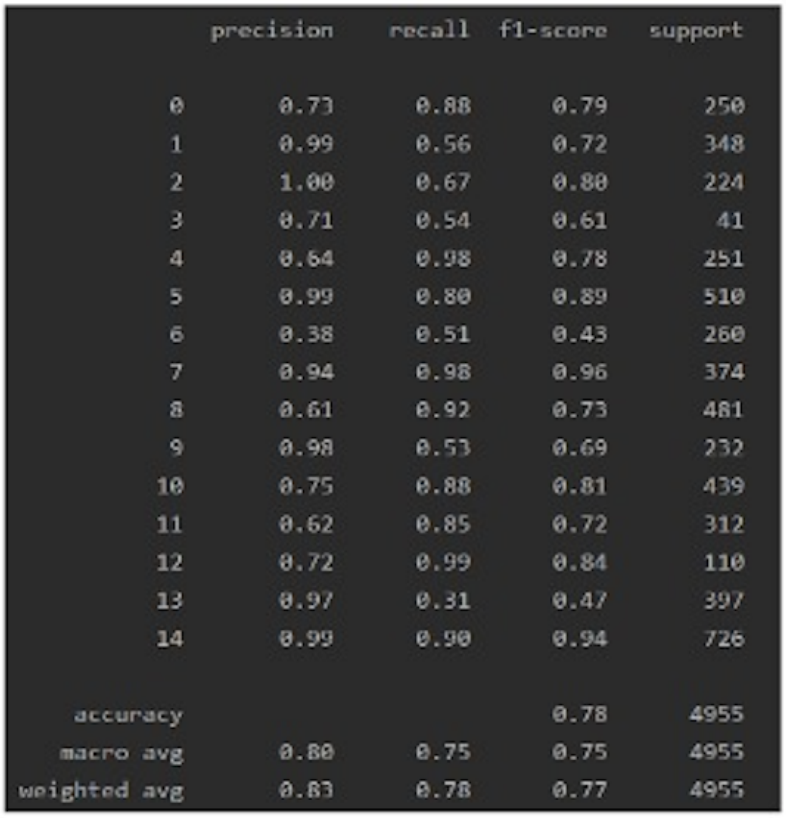
We also need to make a few more settings before we train the model. The following steps need to be taken into consideration:

1. Loss function— It will help us understand how accurate the model is during the training. sparse categorical_cross entropy is used here as we have required multi-class classification tasks and this loss function helps in taking each class in the form of an integer.

2. Optimizer — This tells us how the model will be updated based on the data and the loss functions. We have used Adam optimizer as the optimizer. Adam optimizer is used as it helps in providing a good result by combining the advantages of the two stochastic gradient descents(AdaGrad and RMSprop).

3. Metrics — Helps to monitor the training and testing steps.

The model is later trained for 20 epochs and the accuracy is evaluated. We get an accuracy of around 77%. To give a brief overview of how our model worked, Fig(4) shows the accuracy that the model provided:



	precision	recall	f1-score	support
0	0.73	0.88	0.79	250
1	0.99	0.56	0.72	348
2	1.00	0.67	0.80	224
3	0.71	0.54	0.61	41
4	0.64	0.98	0.78	251
5	0.99	0.80	0.89	510
6	0.38	0.51	0.43	260
7	0.94	0.98	0.96	374
8	0.61	0.92	0.73	481
9	0.98	0.53	0.69	232
10	0.75	0.88	0.81	439
11	0.62	0.85	0.72	312
12	0.72	0.99	0.84	110
13	0.97	0.31	0.47	397
14	0.99	0.90	0.94	726
accuracy			0.78	4955
macro avg	0.80	0.75	0.75	4955
weighted avg	0.83	0.78	0.77	4955

Figure 4: Accuracy for the model created

After training the model for 20 epochs, we create a graph to understand the loss and the accuracy function for the CNN model.

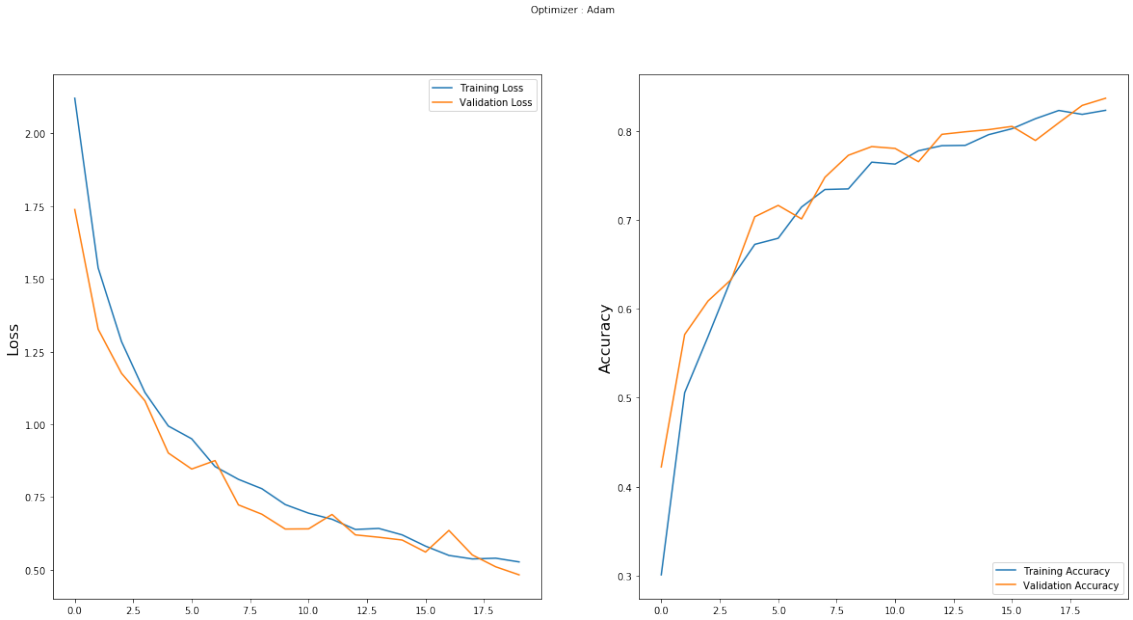


Figure 5: Confusion matrix of the logistic regression

The Fig(5) shown here describes how the loss function and the accuracy are working. The loss function considered explains how with each increasing epoch the loss function is decreasing for both the test data and the train data. While at the same time the accuracy of the model is increasing with each epoch for both the test data and train data-set. Thus we can say that with each epoch the model starts improving and performs better.

We have created a confusion matrix based on the model that we have trained. The confusion matrix helps in providing a tabular format to visualise the performance of the algorithm in supervised learning. The Fig(6) explains the confusion matrix that we got from the logistic regression:

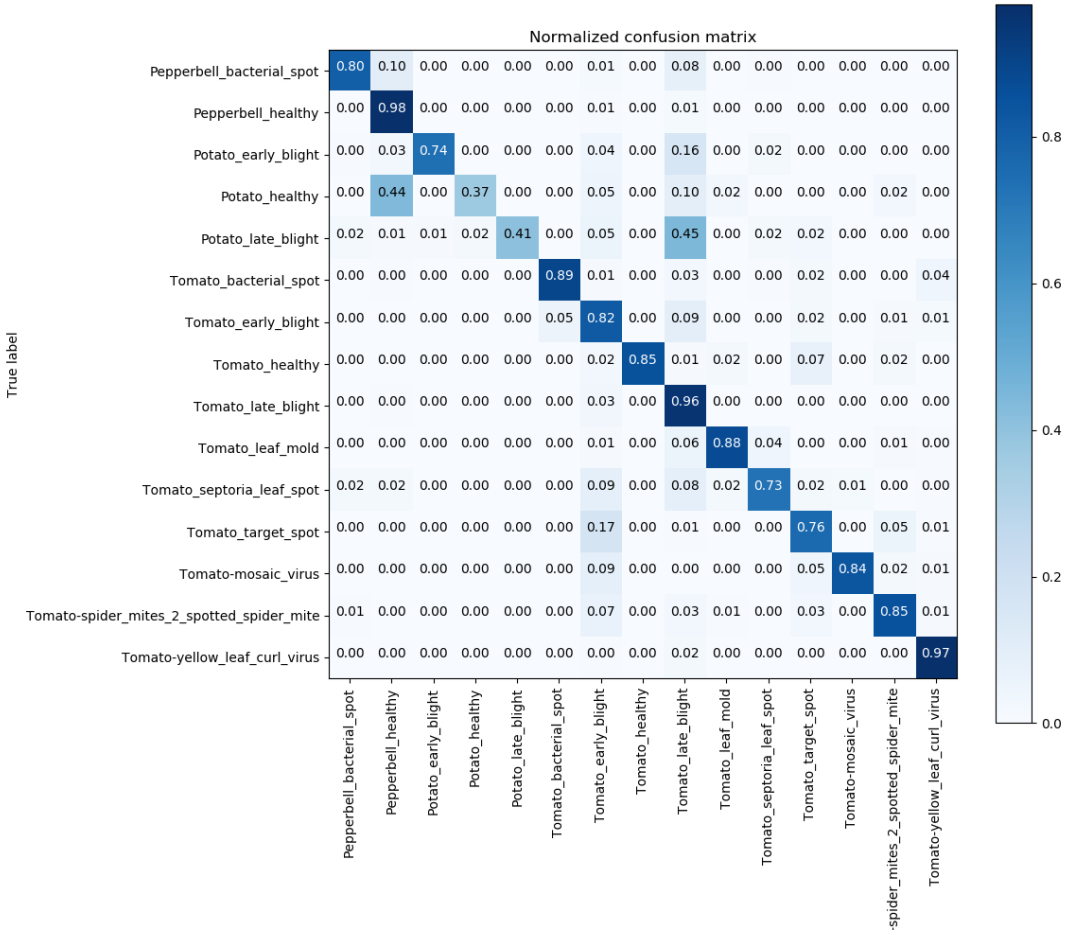


Figure 6: Confusion matrix of the logistic regression

5 Verification of the Models

In this section, we discuss the better model between Logistic Regression and Convolutional Neural Network to justify the choice of our DSS.

As we can already see that the Convolutional Neural Network had better accuracy compared to that of the Logistic Regression model, we can say that we have an edge towards the CNN model to be used for the DSS. The CNN model also has few advantages over the Logistic Regression model. The CNN model works faster on a large amount of data and provides a better and more accurate result. The CNN has an excellent property where the results are independent of the background of the images, as described in the papers (Deep Learning for Tomato Diseases: Classification and Symptoms Visualization, Mo-

hammad Brahmni, 2017). The model solely focuses on the leaves by ignoring the background.

As explained in the paper(Logistic Regression and Convolutional Neural Networks Performance Analysis based on Size of Dataset, Kartik Chopra,2018), the architecture of the CNN has an advantage over logistic regression while used for image classification. In logistic regression, the dependent variables are binary or dichotomous. Thus it is mainly present in the form of 1 or 0. However, the CNN architecture takes care of the 2D structure of an input image. The local connections, tied weights, pooling layers present in the CNN architecture helps in detecting the features correctly from an image, thus providing a higher level of accuracy for the model created. Accuracy of the CNN model also increases due to the various parameters present. The parameters which mainly play a vital role in providing higher accuracy are namely learning rate, batch size etc. CNN models are also much easier to train and it can also have fewer parameters compared to that of the fully connected networks. Being highly scalable, it allows flexible programming.

6 Discussion

In this section, we will make a quick recap of our decision support system and the algorithms we worked on and now we will also discuss new solutions and how they can be implemented in the future.

1. We have constructed two algorithms, the first one based on logistic regression and the second based on convolutional neural network. The accuracy of our models was taken into account for the decision of which algorithm to use on the decision support system. Clearly, the convolutional neural network had better results and accuracy. The network didn't confuse the classes and did the classification well enough, comparing to the logistic model, which used to confuse many of the classes. As we mentioned before our decision support system is based on our convolutional neural network, but we had some limitations to deal with. First, the computer power of our systems didn't allow us to test models that require more training and could have better results. Although the results were actually expected and good enough for our model, we encourage the construction of new models that yield better results.

2. As further research, building new, better convolutional neural networks are recommended. The limitation of computer power is an important constraint. By overcoming this obstacle new techniques can be tried and new, better results can be obtained. New convolutional neural networks with enough power can have the following parameters :

   (a) Using images without resizing them

   (b) Training for more than 20 epochs

   (c) Try to find more data to train and evaluate the model

These can be some reasons that the new model could have better results. As a suggestion to future work an important aspect could be the detection of bacterial spots in the plant leaves, so the user can immediately identify in which leaf and where exactly is the bacterial spot.

7 User's Manual of the DSS

In this section, we will describe accurately how exactly our decision support system works, to minimize the user's errors when using the system. The system consists of four (4) main buttons and many other buttons that open every time in new pop-up windows. A step by step explanation of the buttons will be described and how can be the results found in the folder. An overview can be seen in the following picture :

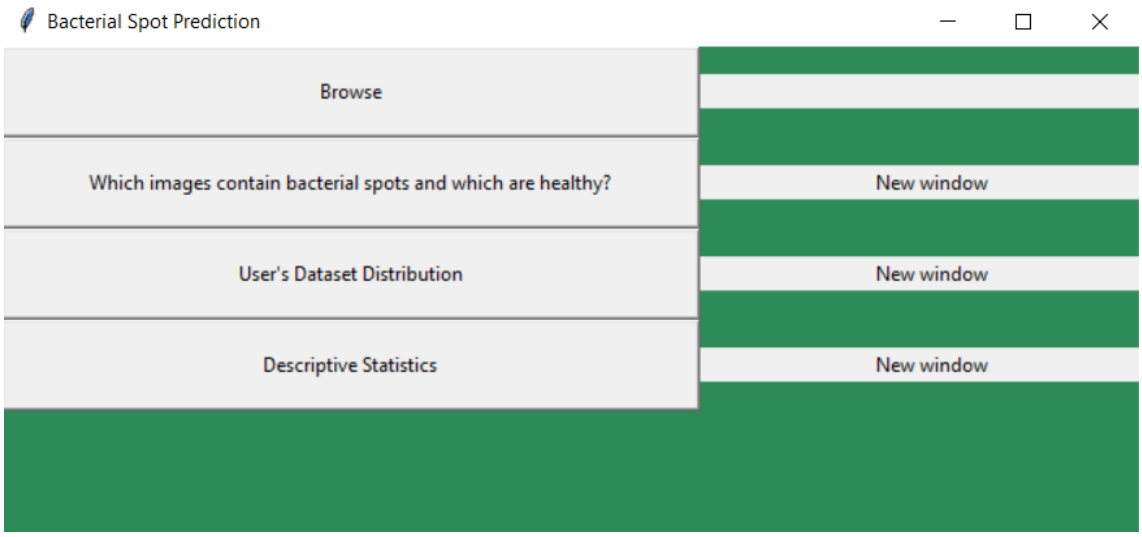
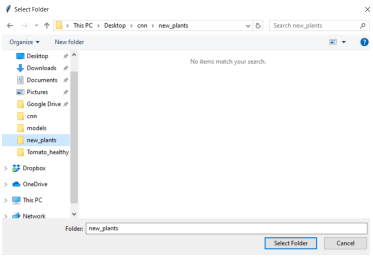


Figure 7: Overview of DSS

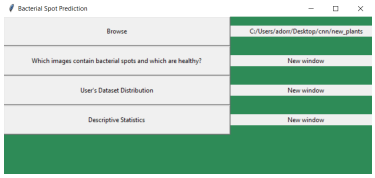
7.1 Button: Browse

This button helps the user in finding and selecting the data-set with images which we use for classification. The user can select the desirable folder and we can see the file location appearing next to the browse button of the interface after we select the desired folder with the date-set present in there. If the path is wrong the user can browse again to find the right path of his dataset.

IMPORTANT WARNING: The sizes of the images of the given path from the user will be converted to 32x32 pixels because the DSS expects images with size



(a) Selecting the folder



(b) Showing the path next to the button browse

32x32. So if the user wants to keep his current folder with images it will be good if he/she can create a copy of the dataset, so the dataset can't be lost.

7.2 Button: Which Images contain bacterial Spots and which are healthy?

This button is considered to be the prediction button. Clicking this button the prediction window will pop up with five (5) more buttons. These buttons are the following:

1. Button: User's dataset length.
   The length of the User's dataset can be understood by clicking on this button

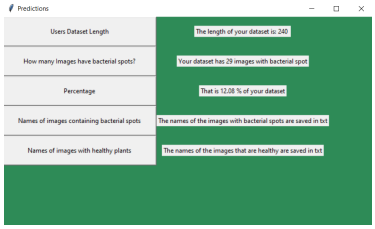
2. Button: How many Images have bacterial Spots?
   By clicking this button, it will show the number of images containing bacterial spots

3. Button: Percentage
   By clicking it shows the percentage of images containing bacterial spots

4. Button: Names of images containing bacterial spots
   By clicking this button, we trigger the DSS to APPEND the names of all the images containing bacterial spots and store it in a text file with the same name.

5. Button: Names of images with healthy plants
   By clicking the names of the images with healthy plants button, the images of the leaves which are healthy are APPENDED in the text file with the same name.

(a) Predictions window



(b) Example of clicking the buttons

7.3 Button: Descriptive Statistics

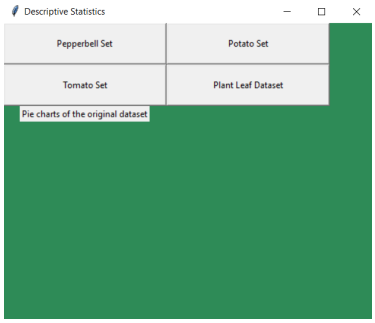
This button is placed in the DSS with an important role. By clicking on this button, a new window pops up with four (4) new buttons. In this section, we have included static pie charts. The static pie charts provide a brief overview of the data present. We get pie-charts separately for the subclasses present within the three main classes and also a pie chart is separately provided to show us the overall division of the leaves within each subclass.We shall note here that the Descriptive Statistics provide the pie charts for the training data set that we already have. The buttons are the following :

1. Button: Pepper bell Set
   It shows the pie chart with percentages from the whole pepper bell set.

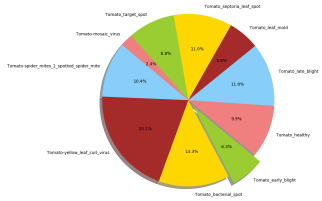
2. Button: Potato Set
   It shows the pie chart with percentages of the whole potato set.

3. Button: Tomato Set
   It shows the pie chart with percentages of the whole tomato set.

4. Button: Plant Leaf Dataset
   It shows the pie chart with percentages from the dataset.



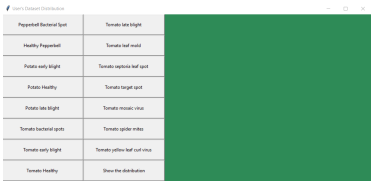
(a) Descriptive Statistics window



(b) Example by clicking in the tomato Set button

7.4 Button: User's Dataset Distribution

This button predicts how the images of the user's dataset are distributed among the fifteen (15) subclasses.Clicking on this button the window with name "User's Dataset Distribution" will pop up with the names of the fifteen (15) subclasses.The user can click on the button "Show the distribution" to see how many images of the user's dataset are distributed in each subclass. An instance can be seen below :



(a) User's dataset distribution window

(b) Example by clicking on the button "Show the distribution"

8 Documentation of the DSS

The programming language that is used to build the DSS is python 3(+). In this section, we will try to explain all parts of the code. Also, functions will be described and why they are used. Functions allow us to be more effective when in the construction of the DSS. In our DSS code, you can find three (3) attached files, all of them will be further explained.

resizing.py

In this file, we have written the code that was necessary to resize the given dataset of the assignment. After that, a new folder was created with the same name and structure, but with the resized images. The images were 256x256 pixels and became 32x32.

resize()

This function the only function of the resizing.py file. It resizes the given dataset.

CNN_model_GUI.py

In this fie the convolutional neural network was built. Pre-built functions from important packages were used here for the architecture of the CNN, the splitting of the data to 70% training data and 30% test data, for compile of our model and the fitting. Keras helped us create the convolution neural network and fitting the data to our model and sklearn was used for splitting the data.

plot_confusion_matrix()

This function was created to visualize in a better way the confusion matrix, a pre-built function in sklearn package that helped us understand the performance of our model. If it classifies the images well to the subclasses or confuses the subclasses among the predicted labels and the true labels.

GUI.py

In this file, you can find the most important functions that were built for our DSS. First, the most valuable package is the tkinter, because based on this package was our DSS constructed.

graph(), graph1() , graph2(), descrstats()

These functions were built based on the package matplotlib and are static functions that provide us with useful percentage piecharts of the original dataset distribution. The function graph() is representing the pepper bell set, graph1() represents the potato set, graph2() represents the tomato set and descrstats represents the whole dataset.

browse_button()

This function was built to help the user reach his dataset-folder and to identify the right path. It reads the path and feeds the images to the DSS after resizing them to 32x32 pixels. This function is called in the DSS by clicking on the button "Browse".

distribution()

This function was built to predict in which of the fifteen (15) subclasses are the user's images distributed in a new pop up window. It loads our CNN model, fits the new dataset that was previously obtained by the browse button and produces the distribution by clicking the button "Show distribution".

predictions()

This function pops up a new window and makes the predictions of the user's dataset with the same way as the function distribution, so there is no confusion, about which button to click on first. It loads our CNN model and produces the user's dataset length, the predicted number of images with bacterial spots, the percentage of bacterial spots of that particular dataset, and APPENDS the names of the plants with bacterial spots into a txt file called "names_bacterial_spots_images" where the user can find all the names of the images containing bacterial spots. Also, it APPENDS to another txt file called "names_healthy_images" the healthy images of the given dataset.

new_window()

This function pops up a new window with four (4) buttons.These buttons are used to call the functions graph,graph1,graph2,graph3 and descrstats.

9 Task Separation

week	task	time
week 1		
Apala	Literature Research,DSS	45
Kostas	Preprocessing the Data	55
week 2		
Apala	DSS, Logistic regression	50
Kostas	Logistic Regression, CNN, Proofreading Draft Report	45
week 3		
Apala	Decision Support System	50
Kostas	Convolutional Neural Network	50
week 4		
Apala	Scientific Explanation of Algorithms, Working on Report	50
Kostas	DSS , Proofreading the Report	55

10 References

1. Logistic Regression and Convolutional Neural Networks Performanc Analysis based on Size of Dataset, Kartik Chopra, Srimathi. 2018 IJEDR

2. Deep Learning for Tomato Diseases: Classification and Symptoms Visualization, Mohammed Brahimi,2017

3. Logistic Regression and Convolutional Neural Networks Performance Analysis based on Size of Dataset, Kartik Chopra,2018

4. A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition,Alvaro Fuentes.

5. https://towardsdatascience.com/from-raw-images-to-real-time-predictions-with-deep-learning-ddbbda1be0e4