By Jubin Mohanty

Plant Species Classification using V2 Plant Seedling Dataset (Kaggle)

Convolutional Neural Network

Domain: Computer Vision

Capstone Project

Machine Learning Engineer Nanodegree, Udacity

2nd July 2021

Table of Contents

Definition

Project Overview

As the University of Pretoria stated, the successful cultivation of crops broadly depends on weed control strategies. It has been widely observed that tons of cultivated crops are wasted due to crop infestations. A zero percent loss in crops is entirely a rare event; generally, there are 10 to 100% losses as per the current weed control practices. (Pretoria, n.d.)

Additionally, for farmers, it is of vital importance to detect weed in the initial first to six weeks of plantation because both weed and crop vigorously search for nutrients and water in the soil for this specific period.

According to the research paper published by Thomas Mosgaard Giselsson and co, there is no robust computer vision system out there that can classify ground-based weed species. (Giselsson, Jørgensen, Jensen, Dyrmann, & Midtiby, 2017)

In this project, I have developed a CNN model which can detect plant species with up to 92% of accuracy.

Problem Statement

Here, I am using a dataset that contains 5,539 images of crop and weed seedlings. The images are grouped into 12 classes. Moreover, these classes represent common plant species from Danish Agriculture. (Dyrmann)

Each class contains RGB images, which reflect various stages of plant growth.

Using this dataset, the goal is to build a model to further classify weed seedlings and crops. Additionally, to solve this classification problem, I used a convolutional neural network architecture, where I used a pre-trained model and few custom layers to build the model.

Moreover, I have also planned to further extend this project by integrating this Deep Learning API with a web or mobile application, where a farmer will have to upload the image in the mobile or web application, and the API will be able to predict the species of the weed or crop. Thus, a farmer can take appropriate action. In this way, if a farmer found a specific seedling as a weed, it can be destroyed before it infested the actual crop.

The actual data set can be found out at the link mentioned below.

https://vision.eng.au.dk/plant-seedlings-dataset/

Metrics

The development of machine learning models revolves around the idea of the constructive feedback principle. When we build a model, the evaluation metrics are used to get feedback, and thus further improvement is continued to achieve the desired accuracy. (Srivastava, 2016)

Simply building a predictive model is not the prime motive; we must utilize the capability of evaluation metrics to discriminate among model results.

In this project, the evaluation metrics that I have used to validate the model and improve model accuracy are Validation Accuracy, Precision, Recall, F-1 score, and Confusion Matrix.

- Validation Accuracy is where both true positive and true negative is taking into consideration.

$$\text{Validation Accuracy} = \frac{true\ negatives + true\ positives}{size\ of\ the\ dataset}$$

- Precision: It tells the accuracy of positive predictions.

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}$$

- Recall defines the fraction of positive cases that the model is able to catch or correctly identified.

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

- F1 scores tells the percent of correct positive predictions. It is nothing but the weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst score is 0.

$$F1\ Score = \frac{2*(Recall*Precision)}{(Recall+Precision)}$$

Moreover, in this classification problem F1 score is preferred to be the most important metrics because it takes account of both false positive and false negative. Originally this data set has uneven class distribution, therefore, F1 score will be more useful than accuracy.

ANALYSIS

Data Exploration

The dataset consists of around 5,539 images and weed seedlings. All the images are further classified into 12 groups or classes.

These classes are nothing but the common plant species from Danish Agriculture.

Image Types: All the classes contain RGB images in .png format having different sizes.

Below is a list of all the 12 classes:

```
['Scentless Mayweed',
 'Common wheat',
 'Charlock',
 'Black-grass',
 'Sugar beet',
 'Loose Silky-bent',
 'Maize',
 'Cleavers',
 'Common Chickweed',
 'Fat Hen',
 'Small-flowered Cranesbill',
 'Shepherd's Purse']
```

Moreover, as an initial data exploration step, I want to investigate the count of images present in each class such that I can examine whether the dataset is balanced or not.

```
Scentless Mayweed: 607
Common wheat: 253
Charlock: 452
Black-grass: 309
Sugar beet: 463
Loose Silky-bent: 762
Maize: 257
Cleavers: 335
Common Chickweed: 713
Fat Hen: 538
Small-flowered Cranesbill: 576
Shepherd's Purse: 274

Total Images:- 5539
```

Image shape is in 3 dimensions. Below is the shape of a randomly chosen image.

```
(330, 330, 3)
```

Exploratory Visualization

First and foremost, I want to plot images in a clean, tight grid-like structure, along with its class name(species). The approach which I adopted was to first create a pandas data frame with filename and target.

Note: Here, the filename denotes the path of the image, and target is the species name.

Using this strategy, I would set up the data for further preprocessing, where it will be further balanced and split into train and test sets.

	file_path	target
0	./all_files/Scentless Mayweed_511.png	Scentless Mayweed
1	./all_files/Maize_182.png	Maize
2	./all_files/Scentless Mayweed_277.png	Scentless Mayweed
3	./all_files/Sugar beet_13.png	Sugar beet
4	./all_files/Small-flowered Cranesbill_479.png	Small-flowered Cranesbill
5	./all_files/Small-flowered Cranesbill_312.png	Small-flowered Cranesbill
6	./all_files/Fat Hen_54.png	Fat Hen
7	./all_files/Common wheat_241.png	Common wheat
8	./all_files/Cleavers_79.png	Cleavers
9	./all_files/Shepherd's Purse_50.png	Shepherd's Purse

Fig1. Denotes a pandas dataframe with filename and target as label.

Finally, I plot the images in a grid format having 3 rows and 5 columns, where I labeled each image with its species name.

Fig2 is a plot of 25 images having its species as its label.

For Data balancing, I first performed some exploratory data analysis and visualization to fetch information about the overall data distribution.

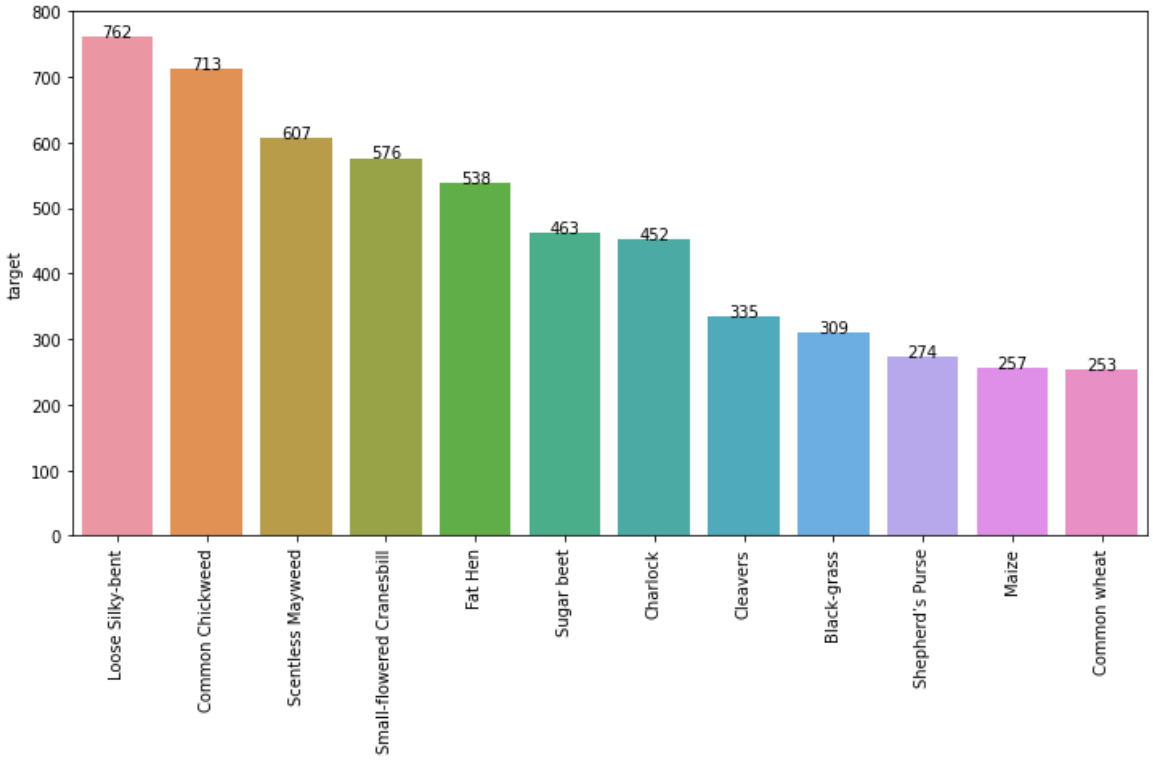

Fig3 is a bar plot, which display the information regarding the counts of all images categorized based on species type.

From the above figure, it is quite evident that the data distribution is quite contrasting in nature, where Loose Silly-bent has 762 images compared to common wheat, which is only 253.

This specific situation will be highly problematic in a classification scenario, as it will lead to misleading accuracy of classes. To solve this issue, I have equally distributed the images for each class. Because of this, now each class will have the necessary amount of information for training a robust and accurate model.

Also, another vital aspect of a neural network is that while building the architecture of CNN, we have to decide the layers of CNN based on the input size of images. Also, the images should have the same size while feeding to the input layer.

Thus, it is always good to resize the images, especially to a smaller size, before feeding them to CNN.

Since we have to investigate the size of images, therefore I performed a jointplot of the seaborn library, which basically plots the dimensions (height & width) of all the images in the x and y-axis.

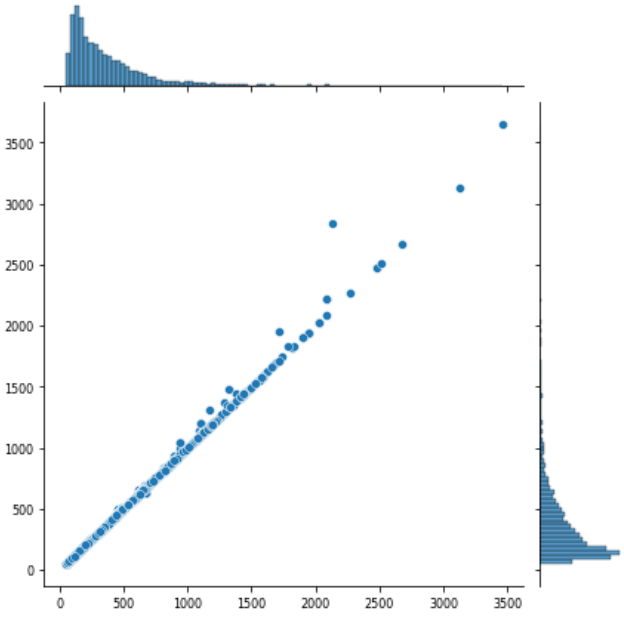

Fig4 gives information about the shape of the image ranging from 0-500 to >3500, therefore in preprocessing steps we will resize the images.

Algorithms and Techniques

When it comes to deep learning, a convolutional neural network (CNN or ConvNet) is a class of deep neural network, which is commonly used to analyze visual imagery. (Convolutional neural network, n.d.)

A convolutional neural network is primarily inspired from biological neurons, which have a connectivity pattern that resembles an animal visual cortex. It has an architecture that consists of an input layer, hidden layers, and an output layer. In any feed-forward network, middle layers are hidden because the activation function masks their input and outputs. Also, the convolutional neural network has various

layers such as kernels(filters), pooling layers, where feature extraction takes place before a fully connected deep neural network for classification.

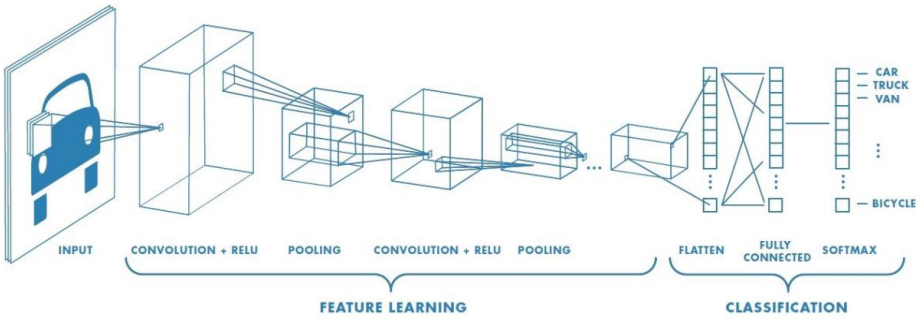

Fig4a: showing CNN layers and fully connected layers for classification.

Benchmark

As a benchmark for this classification model, I followed Kaggle's evaluation metrics. As per Kaggle, for this competition project, meanFScore, which is actually a micro-averaged F1-score, is considered as a benchmark. It is nothing but the harmonic mean of precision and recall. (Plant Seedlings Classification, n.d.)

Moreover, I took the leaderboard score as a benchmark which is 0.99. Also, while developing the CNN architecture, I initially secured 70-80% F1 score by using only custom layers. Therefore, my goal was to tune the hyperparameters and make necessary changes in the algorithm to increase the model F1 score to at least 90%.

Additionally, I face few challenges in execution time for training the model. Initially, it was taking more than 30 minutes with the VGG19 model. Thus, my target was to reduce the execution time to not more than 10 minutes.

Methodology

Data Preprocessing

The steps which are involved in the data processing are mentioned below:

1. First, I copied all the images into a file, and then according to each class(species), I wrote a logic, which balances the entire dataset.

	file_path
target	
Black-grass	250
Charlock	250
Cleavers	250
Common Chickweed	250
Common wheat	250
Fat Hen	250
Loose Silky-bent	250
Maize	250
Scentless Mayweed	250
Shepherd's Purse	250
Small-flowered Cranesbill	250
Sugar beet	250

Fig6: This is how the dataset looks like after balancing.

2. Now, I created a directory that contain further two subdirectory's called to train and valid.
3. Using the scikit learn train test split method, I split the dataset (Fig6) into train and validation.
4. After splitting, I reshaped all the images where all of them are set to equal width and height and using the opencv imwrite method, I copied all the images into their respective train and test subfolder. Here, the main idea is to allow ImageDataGenerator to load the images from these subdirectories. Thus, I have followed this specific folder structure to save image files.

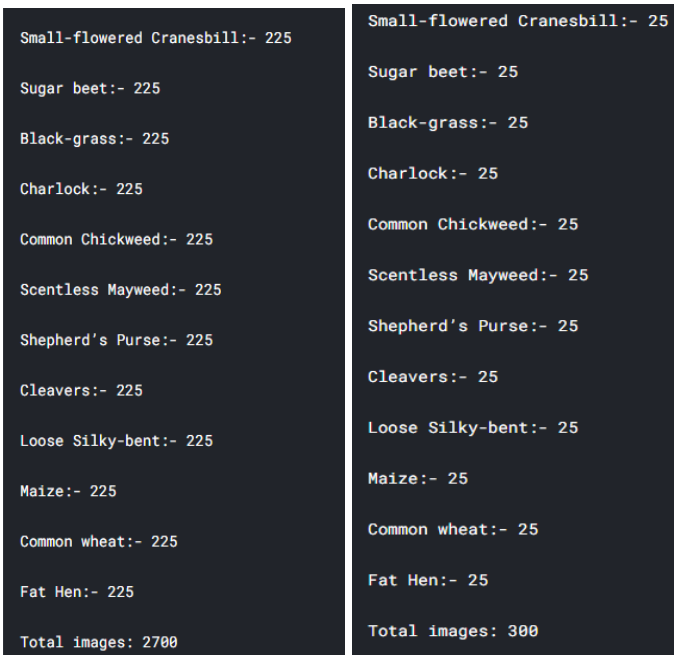

Fig7: The figure display the count of images in train and validation folder.

Implementation and Refinement

So, the implementation is divided into four parts:

1. Data Augmentation.

2. Data Loading.
3. Developing CNN model.
4. Training the model.

- In the **data augmentation** step, I used the Keras ImageDataGenerator class to artificially expand the training dataset to create modified versions of the image. In this way, it will improve the performance of the model to generalize.
  However, we still need to be careful while making modification because it might also lead to extra noise in the image and can substantially decrease the accuracy of the model.

```
datagen = ImageDataGenerator(rotation_range=30, # rotate the image 30 degrees
                             width_shift_range=0.2, # Shift the pic width by a max of 2%
                             height_shift_range=0.2, # Shift the pic height by a max of 2%
                             shear_range=0.15, # Shear means cutting away part of the image (max 15%)
                             zoom_range=0.15, # Zoom in by 15% max
                             horizontal_flip=True,
                             fill_mode="nearest", # Fill in missing pixels with the nearest filled value
                            )
```

- In **the data loading** step, I have used the flow_from_directory() method of ImageDataGenerator class. It will basically load images to train, validation and test object from the respective class(species) folder containing images, which is further used while training, and prediction.
- While developing the CNN model, I used an InceptionResNetV2 as my pre-trained mode, which then further feed to a hidden dense layer with 128 layers, and finally to the output layer consists of 12 layers after flattening the pre-trained model.

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
image_input (InputLayer)     [(None, 96, 96, 3)]       0
_________________________________________________________________
inception_resnet_v2 (Functio (None, None, None, 1536)  54336736
_________________________________________________________________
flatten_2 (Flatten)          (None, 1536)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               196736
_________________________________________________________________
dense_5 (Dense)              (None, 12)                1548
=================================================================
Total params: 54,535,020
Trainable params: 54,474,476
Non-trainable params: 60,544
_________________________________________________________________
```

Activation Function

- For the hidden layer, I have used rectified linear activation function, as it is quite simple to implement, and less susceptible to vanishing gradients that prevent deep learning model from being learned.

- In case of the output layer, I used Softmax as an activation function because it is a case of mutually exclusive classes. Thus, here the range will be 0 to 1, which are nothing but the probabilities of each class.

Cost Function

- In CNN after each epoch the network try to learn and iterate through different weight value to minimize the error.
- The best approach for this would be Adam optimizer, and this is what I have used in my network. In this way, during the backpropagation, the network is updated with a new weight.
- This optimizer initially starts with larger steps, and once the slope gets closer to zero, it tends to take smaller steps.
- Then, I compiled the model with I compile the model, with a loss parameter assigned to categorical_crossentropy as it is a multi-class classification problem to calculate loss.

- In the last step, I **train** the model, where I pass training tensor data, epochs, steps per epoch, validation data and steps, and callbacks as its parameter values.

Results

Model Evaluation and Validation

After a lot of iteration and tuning the hyperparameters, I come up with the above-mentioned CNN architecture.

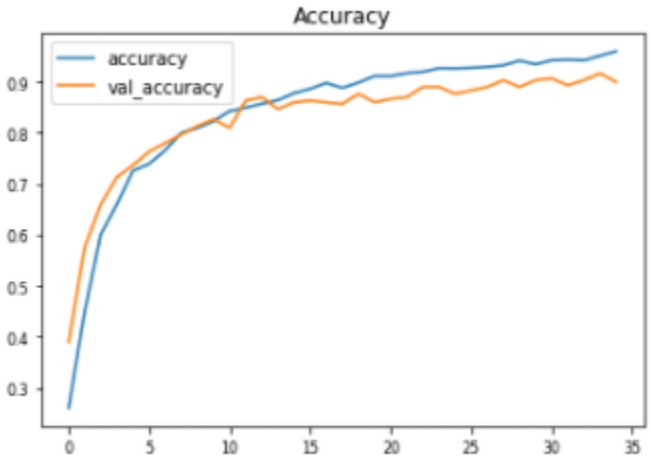

Fig 8 Line Plot

As per the above graph, it is quite evident that over the period of 35 epochs, both accuracy of the training and validation model significantly improved.

Also, the validation accuracy is about 92%, which is better than what I set it as a benchmark.

```
Validatio Loss: 129.3146
Validation Accuracy: 92.33%
```

Moreover, I used precision, recall, and F-1 scores to rate the model accuracy. For this, I generate a classification report which explains the prediction accuracy of each class individually.

	precision	recall	f1-score	support
Black-grass	0.70	0.64	0.67	25
Charlock	0.89	0.96	0.92	25
Cleavers	0.96	1.00	0.98	25
Common Chickweed	1.00	0.92	0.96	25
Common wheat	1.00	0.96	0.98	25
Fat Hen	1.00	1.00	1.00	25
Loose Silky-bent	0.68	0.76	0.72	25
Maize	1.00	1.00	1.00	25
Scentless Mayweed	1.00	0.96	0.98	25
Shepherd's Purse	0.93	1.00	0.96	25
Small-flowered Cranesbill	0.96	0.96	0.96	25
Sugar beet	1.00	0.92	0.96	25
accuracy			0.92	300
macro avg	0.93	0.92	0.92	300
weighted avg	0.93	0.92	0.92	300

Additionally, I am making a prediction for a set of 25 test images to know the true positive, true negative, false positive, false negative. I generated a confusion matrix.

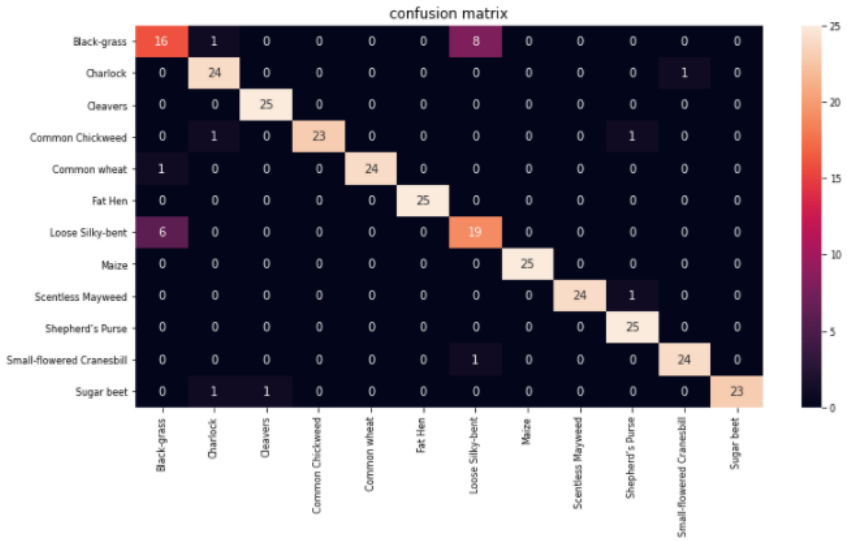

Fig 9: Confusion Matrix seaborn heatmap

As this is a Kaggle competition project, I finally created a CSV file for submission, which has one column contains the name of the image and another column contains the predicted class(species).

	file	species
0	Black-grass/118.png	Loose Silky-bent
1	Black-grass/161.png	Black-grass
2	Black-grass/175.png	Black-grass
3	Black-grass/180.png	Loose Silky-bent
4	Black-grass/181.png	Black-grass
5	Black-grass/189.png	Black-grass

Justification

According to the Benchmark section, my initial goal was to improve my model F1 score to at least 90% or get as close to the leaderboard, which I successfully achieved.

From the classification report, I improved my initial model and increased the F1 score from 73% to 92%.

Additionally, the initial execution time with VGG19 was more than 30 minutes. However, after introducing InceptionResnetV2 a pre-trained model and using only 128 layers in the penultimate hidden layer, I successfully decreased the execution time for training the model to around 10 minutes.

The current model is highly accurate; however, there is further room for improvement by tunning data augmentation parameters and hyperparameters.

This project also has a real-life application to help farmers identify seed and crop species. In this way, farmers can protect crops before infestation from weed.

Improvements

After experimenting with VGG19 and InceptionResnetV2, I found out CNN will require powerful GPU processors to run the model efficiently, especially while working with computer vision problems due to high-resolution images and big datasets.

An alternate model with which we can go for is MobileNetV2 as our pre-trained model because it is quite light as compared to other models, and it is primarily designed to run on mobile devices to offer better performance.

References

Convolutional neural network. (n.d.). Retrieved from wikipedia: https://en.wikipedia.org/wiki/Convolutional_neural_network

Dyrmann, M. (n.d.). *COMPUTER VISION AND BIOSYSTEMS SIGNAL PROCESSING GROUP.* Retrieved from https://vision.eng.au.dk/plant-seedlings-dataset/

Giselsson, T. M., Jørgensen, R. N., Jensen, P. K., Dyrmann, M., & Midtiby, H. S. (2017, Nov 16). *A Public Image Database for Benchmark of Plant Seedling Classification Algorithms.* Retrieved from https://www.researchgate.net/publication/321095442_A_Public_Image_Database_for_Benchmark_of_Plant_Seedling_Classification_Algorithms

Plant Seedlings Classification. (n.d.). Retrieved from kaggle: https://www.kaggle.com/c/plant-seedlings-classification/overview/evaluation

Pretoria, U. O. (n.d.). *Important Weeds in Maize*. Retrieved from https://www.up.ac.za/sahri/article/1810372/important-weeds-in-maize

Srivastava, T. (2016). *11 Important Model Evaluation Metrics for Machine Learning Everyone should know*. Retrieved from https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/