Plant Classification Using Leaves Recognition

Ahmad Jawaad Shah[1]
School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal, Westville Campus,
Durban 4000 RSA
[1]218029400@stu.ukzn.ac.za

Abstract. Botany is a scientific domain that studies plants, their genetics, ecology, classification and structure. The classification of plants remains an essential component to botanists and those working in tourism, such as tour and field guides. The need to classify plants swiftly and efficiently gives rise to the gap this study seeks to fill. With the advent and improvements in machine learning and computer vision, developing a model that can classify plants autonomously is possible. This study proposes a novel system for plant classification using leaf recognition. In particular, various approaches for image preprocessing, feature extraction and image classification will be implemented and evaluated.

Keywords: Machine learning, image preprocessing, feature extraction, classification

1. Introduction

Plants play a significant role in our daily lives. They provide us with oxygen, food as well as pleasing aesthetics. In scientific domains, plants are used for various tasks. In particular, the medical domain uses plants for the creation of herbal medication[1]. Plants are also widely popular in the tourism industry, especially in destinations such as game reserves, where they are used by tour guides to explain the surrounding habitat.

Through botany, it is possible to identify various species of plants. However, these may come at the expense of time as they may need to be analyzed more closely. This tedious task provides an entrance point for machine learning and computer vision. This study will leverage appropriate techniques and models to design and develop a recognition system for plant classification through the advances in the two aforementioned fields.

Prior work in plant classification includes results documented in [2], [3] and [4]. Many different approaches have been explored to undertake the identified problem. These include variations in selected features as well as experimentation on different classification models. Compared with other methods, such as cell and molecule biology methods, classification based on leaf image is the first choice for plant classification[2]. The motivation behind this choice stems from the fact that leaf images are far more accessible than other plant components. Furthermore, it is easier to transfer and work with leaf images on a computer[2].

Digital image classification can be done through supervised or unsupervised learning. Both of the aforementioned techniques use an automated quantitative decision-making mechanism[6]. Two particularly interesting classification models are Support vector machine classification and deep learning models. Support vector classifiers fall under the family of multi-purpose schemes called Support vector machines(SVM)[6]. In particular, this approach operates by finding a hyperplane in the space of possible inputs.

Deep learning solutions can be implemented through neural networks. One such type of neural network is a Muli-Layer Perceptron neural network. Multi-Layer Perceptron(MLP) networks have been frequently used for image classification. These neural networks are desirable for supervised classification as they don't make assumptions on the data as done by conventional statistical classification approaches[5]. The main concern with MLP models is the selection of appropriate model parameters. The parameter selection process requires a subjective and challenging selection of learning algorithm properties[5]. These will have a direct impact on the model's output classification and need to be chosen wisely. An alternative to an MLP is to use convolution neural networks(CNN). For the purpose of this study, CNN's will not be discussed as they were far too resource-hungry to be comparatively experimented on.

The general structure surrounding the creation of a digital image classification model contains many phases. The various phases are followed sequentially to form a pipeline. The pipeline followed is shown in Figure 1. Segmentation was conducted before preprocessing because the raw input images were clear enough to define a region of interest(ROI). Regarding the feature extraction phase, multiple regional descriptors, including simple and textural descriptors, will be used to characterize images. Furthermore, three supervised classification models will be designed and evaluated. Multi-Layer Perceptron, Support Vector Classifier with a polynomial kernel and a Linear Support Vector Classifier are the three targeted models.

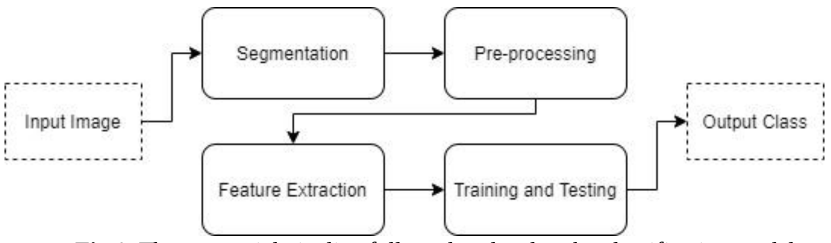


Fig 1. The sequential pipeline followed to develop the classification model. Segmentation was done first because the raw image was clear enough to define a region of interest(ROI).

2. Literature Review and Related Works

2.1 Plant Leaf Recognition Using Texture and Shape Features with Neural Classifiers

Many techniques exist to preprocess and enhance images. Through these techniques, it is possible to use the enhanced images to extract meaningful features. In [3], a novel methodology for characterizing and recognizing plant leaves is proposed. In particular, they seek to use texture and shape features to characterize an image.

Shape and textural features are calculated separately. In the textural preprocessing phase, a Gabor filter is used. This filter is defined as the product of a Gaussian kernel and a complex sinusoid[3]. The remainder of the preprocessing phase uses the grey-level co-occurrence matrices to compute features. The resultant textural feature vector obtained is length 44, as it computes 11 features for each of the four angles in the GLCM.

To form the shape feature vector, the invariant moments and a fast curvelet transform are used. Impressively, the combination of the texture-based and shape-based features can be used to classify images well. It covers many different aspects, including the shape of the leaf and characterizing the texture of the leaf.

Various classification models are available for researchers to use. However, [3] proposes to use a neural model for classification. The primary advantage of using a neural model such as MLP is that classification is done solely based on the number of observations used to train a 'black box' with associated weights[3]. A better approach to the aforementioned model is to use a neuro-fuzzy classifier. The motivation behind such a model stems from the fact neural models such as MLP will return one classification for a data point.

If classes have similar shape and texture, it would be better to receive probabilities instead of exclusive classification. This is where fuzzy neuro models succeed.

2.2 LeafSnap: A Computer Vision System for Automatic Plant Species Identification

In [4], a comprehensive system is developed for plant classification. Utilizing machine learning and computer vision, the model performs well as has been deployed as a mobile app.

In particular, the methods used to achieve this state of the art system are much to be admired. The motivation behind such an advanced system is to assist individuals from all walks of life. These include scientists, botanists, students and many more.

Segmentation of images was done by separating the foreground and background portions of the image, similar to any segmentation process. However, the major difference between [4] and other studies is that segmentation is done in the HSV colourspace. The original colour is converted to the HSV colour space, and approximate segmentation values are computed. These are then refined to remove false-positive regions.

Analysis of the phase above shows that the images are well segmented. The major issue found when segmenting leaves was the overcasting shadows that may have hindered segmentation.

Features used in [4] were curvature-based. These are not standard, easy to compute features. In particular, they typically use differentials, which amplify noise, are sensitive to the orientation and scale of captured images, and are difficult to measure at multiple scales[4]. However, in [4], integral measures are used as they are faster.

More in line with this study, [2] proposes textural features that can be extracted from the grey co-occurrence matrix. Furthermore, a second approach is also proposed, which uses PCA. This approach considers a small feature characterization and seeks to reconstruct the remainder.

3. Leaf Segmentation

Segmentation formed the first step of the pipeline described in Figure 1. In particular, two operations, namely image cropping and image thresholding were conducted on the input images to define a region of interest. Between the cropping and thresholding functions came a grayscale preprocessing operation. This is one preprocessing task that was used in combination with the segmentation process. Once a region of interest was defined, the image was further rescaled to take the dimensions of a 400x400 image. Image scaling was done for uniformity so that the feature extraction process would not be biased to images of different sizes. Figures 3.1 and 3.2 show the output from the cropping and thresholding phases; furthermore, the images' colour distribution is shown, which guided the threshold value selection process.

3.1 Image Cropping

Image cropping was done to remove unnecessary detail from the image. In particular, each image had colour scales at the bottom and right-hand side positions. These needed to be removed before being converted to grayscale. The effect of not removing these scales would result in large areas of dark colours, which would hinder our ROI extraction. For each image, the region to be discarded was calculated dynamically. This can be described as follows:

$$\begin{cases} y = y - 140 & if\ y < 700 \\ y = y - 155 & if\ y < 770 \\ y = y - 180 & otherwise \end{cases} \quad \ldots (3.1)$$

3.2 Image Thresholding

Image thresholding took two forms in this study. For both thresholding functions, the inverse was used so that the resultant region of interests would be a white area.

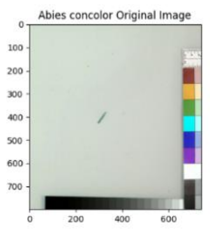
The first thresholding function was the simple global thresholding. For each pixel, if its value was lower than the threshold, it was set to white, and if its value was greater than the threshold, it was set to black. This threshold was useful for many dark coloured leaf images but fell short when the leaf was a light colour. Figure 3.2 is an example of a light colour leaf that struggled to be segmented into foreground and background regions with a global threshold. The threshold function can be described mathematically as follows:

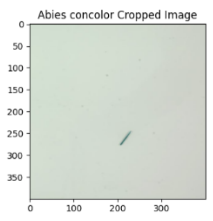
$$\begin{cases} 0\ if\ img[x][y] > 175 \\ 255\ otherwise \end{cases} \quad \ldots (3.2)$$

Due to the downfall of having a fixed threshold value, Otsu's binarization was used as an alternative. This technique determines a threshold value automatically. Otsu's binarization worked well on many images; however, it was better to use the initial thresholding with a predefined value in some instances. This was identified when images such as Figure 3.1 were being binarized, and a large amount of noise appeared in the top right corner.

To counter the thresholding debacle, a mixture of both techniques was used. Initially, each image went through the global thresholding, and if they met certain criteria, i.e. they had a large enough surface area, the image was sent for Otsu binarization.

The output of the segmentation phase resulted in a 400x400 image with the foreground and background visible. These images can now undergo preprocessing and be prepared for feature extraction.





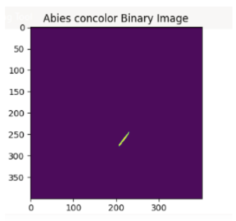
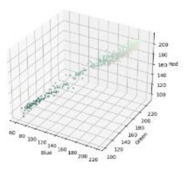


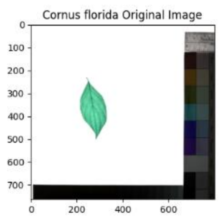
Fig 3.1

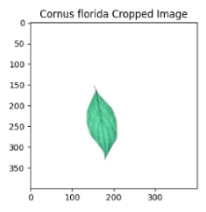
(a) Original **(b) Cropped** **(c) Binarized**



(d) Colour distribution

Fig 3.1. Images from the Abies Concolor class, where global thresholding worked well





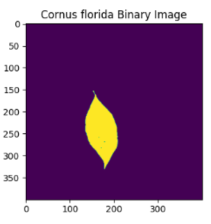
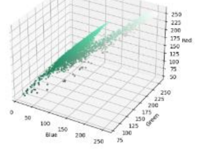


Fig 3.2

(a) Original **(b) Cropped** **(c) Binarized**



(d) Colour Distribution

Fig 3.2. Images from the Cornus Florida class where Otsu Binarization worked better.

4. Leaf Image Preprocessing and Enhancement

Image preprocessing and enhancement formed the second step of the classification pipeline. The images present in the dataset were of good quality and did not need histogram equalization. In particular, four preprocessing techniques were used to obtain the final image. These are conversion to grayscale, thresholding, filtering and morphological operations. The grayscale conversion was used in the segmentation phase, through which the ROI was obtained via thresholding. Figure 4.1 and 4.2 show the sequential steps conducted in the preprocessing of an image

4.1 Grayscale

Each input image was converted to grayscale. This was done using a function provided by OpenCV[7].

4.2 Thresholding

Thresholding was done on the grayscaled images. Furthermore, the operation was conducted using the formula (3.2) from the previous section. Through thresholding, there may still be some noise present in the images. These are handled by filtering.

4.3 Filtering

Filtering was done to remove noise from the binarized image via a median filter of size 3x3. Not all the images underwent filtering. This is due to some leaves having a very small surface area. An example of this is in Figure 4.1. Images underwent filtering under the following conditions:

$$\begin{cases} filtering & if\ area > 2000 \\ no\ filtering & otherwise \end{cases} \dots \quad (4.1)$$

Through filtering, images such as Figure 4.1 would no longer be visible. Hence, the area is considered before a filter is applied.

Morphology

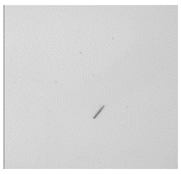
Morphological operations were the last preprocessing technique applied to the images. Similar to the filtering, careful consideration of the area of the images had to be made. Thus, two morphological approaches were used, namely dilation and opening. The conditions for morphology were defined as follows:

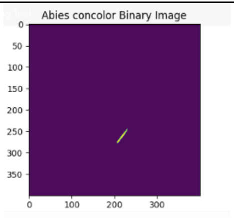
$$\begin{cases} opening & if\ area > 2000 \\ dilation & otherwise \end{cases} \dots \quad (4.2)$$

The motivation behind the application of the opening morphological operator was to remove stems from the leaves. These stems are not necessary to characterize the leaf and can be excluded when representing the image from feature extraction. Filtering would not give the image the same result. If filtering had been used to remove stems, then some boundary data would have been lost.

Dilation. Dilation was used on images that had a small surface area. This was done so that the leaf could become more visible. A 3x3 mask was used for this as it provided a small enhancement. Other masks, such as 5x5 and 7x7 kernels, were not representative of the original image.

Opening. The opening was achieved using a kernel of size 7x7. This kernel effectively removed stems from the leaf without affecting the structure of the other portions of the leaf.





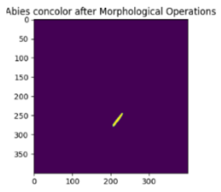
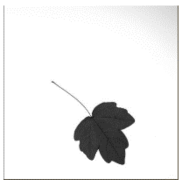
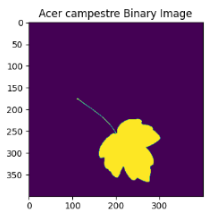


Fig 4.1

(a) Grayscale (b) Binarized (c) Dilation

Fig 4.1. Shows the process of preprocessing an image with an area of less than 2000. The image is converted to grayscale(a), thresholded(b) and finally dilated(c) to make it more apparent.





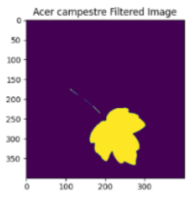
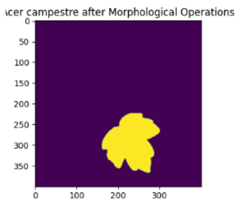


Fig 4.2

(a) Grayscale (b) Binarized (c) Filtered



(d) Opened Image

Fig 4.2. Shows the process of preprocessing an image with an area of more than 2000. The image is converted to grayscale(a), thresholded(b), filtered(c), and finally opened(d) to remove the stem.

5. Leaf Feature Extraction

Feature extraction formed the basis of data collection to feed into the classifiers. In total, twenty-two features were extracted from each image. Regional descriptors were used, and these were both simple and statistically based descriptors. Furthermore, various textural descriptors were used and computed by obtaining a grey level co-occurrence matrix. Each of the features used is described below.

5.1 Area

The area of an image was calculated as the sum of all the white pixels in the image. The colour white represented the foreground of the leaf, and hence the summation returned the cumulative area.

5.2 Perimeter

The perimeter of an image can be defined as the sum of all the foreground pixels having a background pixel as their neighbour.

5.3 Compactness

The compactness can be calculated as the square of the perimeter divided by the area. This provides a measure of how compact the image is.

$$compactness = \frac{perimeter^2}{area} \ldots (5.1)$$

5.4 Convex

We are identifying whether the contour is convex or not. A 0 or 1 is stored in the feature vector. Opencv[7] was used to determine a Boolean value which was then quantified.

$$convex = \begin{cases} 0 & if\ False \\ 1 & if\ True \end{cases} \ldots (5.2)$$

5.5 7 Hu Moment Invariants

These seven features are calculated via a central moment. Six of the seven moments are invariant to translation, scale and rotation. The 7th moment provides a sign change for image reflection[8]. The moments are calculated as follows:

$$h1 = \eta 20 + \eta 02 \quad \text{... (5.3)}$$

$$h2 = (\eta 20 - \eta 02)2 + (2\eta 11)2 \quad \text{... (5.4)}$$

$$h3 = (\eta 30 - 3\eta 12)2 + (3\eta 21 - \eta 03)2 \quad \text{... (5.5)}$$

$$h4 = (\eta 30 + \eta 12)2 + (\eta 21 + \eta 03)2 \quad \text{... (5.6)}$$

$$h5 = (\eta 30 - 3\eta 12)(\eta 30 + \eta 12)[(\eta 30 + \eta 12)2 - 3(\eta 21 + \eta 03)2] + (3.25)(3\eta 21 - \eta 03)(\eta 21 + \eta 03)[3(\eta 30 + \eta 12)2 - (\eta 21 + \eta 03)2] \quad \text{... (5.7)}$$

$$h6 = (\eta 20 - \eta 02)[(\eta 30 + \eta 12)2 - (\eta 21 + \eta 03)2] + 4\eta 11(\eta 30 + \eta 12)(\eta 21 + \eta 03) \quad \text{... (5.8)}$$

$$h7 = (3\eta 21 - \eta 03)(\eta 30 + \eta 12)[(\eta 30 + \eta 12)\,2 - 3(\eta 21 + \eta 03)\,2] - (3.28)\,(\eta 30 - 3\eta 12)(\eta 21 + \eta 03)[3(\eta 30 + \eta 12)\,2 - (\eta 21 + \eta 03)\,2] \quad \text{... (5.9)}$$

5.6 Shannons Entropy

Shannons entropy measures how much information is held in data. In particular, it is the amount of information in an event drawn from the image for an image[9].

$$entropy = -\sum_{k=0}^{L-1} P(r_k) log_2(P(r_k)) \ldots (5.10)$$

5.7 Haralick Features

These features were computed from the grey level co-occurrence matrices. In particular, two GLCM's were computed. One in the horizontal direction(0°) and one in the vertical direction(90°); however, both with a distance of one. From each of these matrices, five of the thirteen Haralick features were extracted. In total, these would make up ten descriptors in the feature vector. The five features extracted from each matrix were contrast, correlation, dissimilarity, energy and homogeneity.

Contrast. The contrast is a measure of intensity contrast between a pixel and its neighbour over the entire image[10].

$$contrast = \sum_{i=1}^{K}\sum_{j=1}^{K}(i-j)^2 p_{ij} \ldots (5.11)$$

Correlation. The correlation is a measure of how correlated a pixel is to its neighbour over the entire image[10].

$$\text{correlation} = \sum_{i=1}^{K}\sum_{j=1}^{K}\frac{(i-m_r)(j-m_c)p_{ij}}{\delta_r\delta_c} \ldots (5.12)$$

Dissimilarity. The measure of dissimilarity is similar to contrast; however, it has a linear dependant off-diagonal of the GLCM.

$$dissimilarity = \sum_{i,j}|\, i-j\,|p_{ij} \ldots (5.13)$$

Energy. Energy is the measure of uniformity in the image.

$$energy = \sum_{i,j}(p_{ij})^2 \ldots (5.14)$$

Homogeneity. Homogeneity measures how similar the pixel intensities contained in the GLCM are.

$$homogeneity = \sum_{i,j}\frac{p_{ij}}{1+|\, i-j\,|} \ldots (5.15)$$

5.7 Feature Normalization

Once all of the features had been extracted, the next step was to normalize the values. Normalization is done to ensure that all of the feature values lie with a specified range. In this study, the min-max feature normalization technique was used. This approach rescales the features to lie in the range [0, 1]. The formula for min-max normalization is defined below:

$$x_{normalized} = \frac{x-x_{min}}{x_{max}-x_{min}} \ldots (5.16)$$

Where x represents the feature being normalized, x_{min} the minimum feature value and x_{max} the maximum feature value for that specific feature column. Table 4.1 and 4.2 describes two feature vectors, without Haralick features due to space constraints, before and after being normalized.

Table 4.1 Unnormalized feature vectors

	Hu 1	Hu 2	Hu 3	Hu 4	Hu 5	Hu 6	Hu 7	Area	Perimeter	Comp-actness	Con-vex	Ent-ropy
V1	3.08	6.55	12.4	13.0	25.7	16.45	26.1	7171	7.65	0.0081	1	0.26
V2	3.17	8.64	10.6	12.7	24.48	-17.19	24.9	12420	501.1	20.22	0	0.39

Table 4.2 Normalized feature vectors

	Hu 1	Hu 2	Hu 3	Hu 4	Hu 5	Hu 6	Hu 7	Area	Perimeter	Comp-actness	Con-vex	Ent-ropy
V1	0.95	0.55	0.41	0.41	0.96	0.925	0.627	0.067	0.000979	0.000055	1.0	0.26
V2	0.99	0.75	0.34	0.40	0.94	0.29	0.61	0.117	0.2424	0.141304	0.0	0.39

6. Leaf Classification

The dataset used for this study was obtained from the LeafSnap[11] website. In particular, the unprocessed lab images was used. Each species of plant had a limited number of leaf images available. This posed a problem as we need the classifiers to see as many images as possible to classify them into their correct classes. Therefore, a split of 90:10 was used. Ninety per cent of the dataset was used for training, and ten per cent was left for testing. This distribution is allocated randomly and handled by SKLearn[12].

The software designed for this study provides three types of training. This is done so that you can decide which portion of the dataset you would want to train and how much. The first option is to train a small portion of the dataset. The number of images used in this case is between 500 and 5000. Typical processing time requires a few minutes.

The second option is to use 5000 to 10000 images. This option takes up to 10 minutes or more (on an Intel i3 with a CPU running at 2.3GHz). Lastly, the third option is to train the full dataset. This can take in excess of 20 minutes when run on the hardware specifications above.

Regarding image classification, a classification model needed to be built and trained. In this regard, three models were identified and implemented. These are Multi-Layer Perceptron, Support Vector Classifier(with the polynomial kernel) and a Linear Support Vector Classifier.

6.1 Multi-Layer Perceptron

The multi-layer perceptron is a neural model and needed careful consideration when choosing the model parameters.

The selected 'solver' for the model was 'lbfgs' as it provides better results when there are few training points. Although the dataset is large, the individual classes have small amounts of images to use. 'Lbfgs' also provided much better results than the 'adam' solver upon comparison. Furthermore, the activation function was set to 'relu' as it allows models to learn faster and perform better. The 'relu' function was compared to the 'logistic' function and provided better results.

6.2 Support Vector Classifier

The standard support vector classifier from SKLearn[12] was used with a polynomial kernel. This was used because the data points may not be linearly separable and should provide better results as compared to the Linear SVC.

6.3 Linear Support Vector Classifier

The linear SVC was used as-is from SKLearn[12]. It is proven to be efficient in learning sparse and dense data. Furthermore, it was included to compare its classification results to that of the SVC with a polynomial kernel.

7. Results and Discussion

The evaluation of the implemented models was carried out by measuring the accuracy, recall, precision, f1-score and hamming loss. Furthermore, in the entire dataset, there are a total of 184 classes. Apart from inter-model comparisons, the models can be compared to the benchmark of assigning a leaf to any class, with a probability 0.0054 of guessing correctly.

The first set of experiments were run on increments of 1000 samples from the dataset. At each point, the accuracy for each of the classifiers was recorded for comparison. This was done for five iterations. Table 7.1 summarises the results.

The second set of evaluations was carried out on the entire dataset. These are documented in Table 7.2. The following subsections discuss the individual metrics and their results.

7.1 Accuracy

The accuracy of the model measures how many test samples the models are able to predict correctly. In this regard, the MLP model outperforms the other two models in every test. Furthermore, the polynomial kernel SVC(PSVC) outperforms the Linear SVC(LSVC). The equation for accuracy can be defined as follows:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \dots (7.1)$$

On the results obtained for the first experiment, it is noticeable that initially, the MLP and PSVC are maintaining good accuracies even though they are decreasing. The takeaway from this experiment is that the LSVC is not the best for image classification because, by the 5000th image mark, the accuracy is nearing 50%. This may be due to the variations in values across the feature vectors and not being able to linearly separate classes well. This is concerning because the entire dataset has not been explored yet. However, the MLP and PSV models are showing promising results.

The MLP model can learn the representations better, and the PSVC can fit the data better than the LSVC. On the entire dataset, the accuracy of the MLP model is 55.46%. This is much better than PSVC(32.53%) and LSVC(28.16). These results are as expected. Judging by accuracy, the MLP model provides the best classification model; however, improvements need to be made to get the accuracy higher.

The accuracy of the model could be increased if there were more training examples for the models to leverage. Nevertheless, other neural models such as CNN and fuzzy neural models can be used and benchmarked against this study MLP implementation.

Table 7.1. Incremental evaluation of the accuracy

	1000	**2000**	**3000**	**4000**	**5000**
MLP	0.92	0.83	0.8267	0.77	0.76
PSVC	0.92	0.75	0.7633	0.695	0.64
LSVC	0.79	0.68	0.637	0.6075	0.552

Table 7.2. Metrics of the full dataset

	Accuracy	**Recall**	**Precision**	**F1-Score**	**Hamming Loss**
MLP	0.5546	0.5546	0.5546	0.5546	0.4454
PSVC	0.3253	0.3253	0.3253	0.3253	0.6747
LSVC	0.2816	0.2816	0.2816	0.2816	0.7184

7.2 Precision and Recall

The precision score is calculated as the ratio of the sum of true positives to the sum of true positives and false positives. Furthermore, the precision value tells you how relevant positive predictions are. A high precision score means that the model is doing well with its classification and implies that there is a low false-positive rate. Final precision rates for the models are not great. These need to be improved to have a model that will be reliable.

The recall value is calculated as the ratio of true positives to the sum of true positives and false negatives. Similarly to the recall, the overall recall is poor. A much better representation would be where one of these two metrics is maximized whilst the other is somewhat in a respectable range.

7.3 F1 Score
The F1-score is the weighted average of precision and recall. As seen from Table 7.1 and 7.2, the f1 score is relatively the same as the precision and recall. Following from the previous section, changes in precision and recall will affect this metric. The F1-score can be calculated as follows:

$$f1 = 2\,(\frac{Precision * Recall}{Precision + Recall}) \ldots (7.2)$$

7.4 Hamming Loss
Contrary to accuracy, the hamming loss measures the fraction of labels that are incorrectly predicted. For the first experiment, the hamming loss is relatively small, but this value increases as the number of classes increases. Again, the MLP provides the lowest hamming loss from the three classifiers. This further indicates that the MLP model is identifying the best from all of the models available.

8. Conclusion
The content of this study covered image preprocessing, image segmentation for the obtainment of a region of interest, feature extraction and the parameters needed to develop and train classifiers.

The overall results from the experiments are satisfactory. It is apparent that the classifiers perform well on a small sample of the dataset, but as the number of classes grows, the accuracy drops. This indicates that the model can be improved with regards to scalability.

From the three classification models, the MLP outperforms the SVC and Linear SVC. Furthermore, the MLP classifier maintains over 50% accuracy, suggesting that neural models are highly effective in image classification.

In terms of feature extraction, future works stemming from this research could seek to add more dimensions to the feature vectors of an image. Furthermore, convolution neural networks could be used and benchmarked against this system.

9. References

[1] F. Jamshidi-Kia, Z. Lorigooini, and H. Amini-Khoei, "Medicinal plants: Past history and future perspective," *Journal of Herbmed Pharmacology*, vol. 7, no. 1, pp. 1–7, Jan. 2018, doi: 10.15171/jhp.2018.01.

[2] A. Ehsanirad, "Plant Classification Based on Leaf Recognition Plant Classification Based on Leaf Recognition," *IJCSIS) International Journal of Computer Science and Information Security*, vol. 8, no. 4, 2010, Accessed: Jun. 25, 2021. [Online].

[3] J. Chaki, R. Parekh, and S. Bhattacharya, "Plant leaf recognition using texture and shape features with neural classifiers," *Pattern Recognition Letters*, vol. 58, pp. 61–68, Jun. 2015, doi: 10.1016/j.patrec.2015.02.010.

[4] E. Adetiba *et al.*, "LeafsnapNet: An Experimentally Evolved Deep Learning Model for Recognition of Plant Species based on Leafsnap Image Dataset," *Journal of Computer Science*, vol. 17, no. 3, pp. 349–363, Mar. 2021, doi: 10.3844/jcssp.2021.349.363.

[5] G. M. Foody, "Supervised image classification by MLP and RBF neural networks with and without an exhaustively defined set of classes," *International Journal of Remote Sensing*, vol. 25, no. 15, pp. 3091–3104, Aug. 2004, doi: 10.1080/01431160310001648019.

[6] T. Oommen, D. Misra, N. K. C. Twarakavi, A. Prakash, B. Sahoo, and S. Bandopadhyay, "An Objective Analysis of Support Vector Machine Based Classification for Remote Sensing," *Mathematical Geosciences*, vol. 40, no. 4, pp. 409–424, Mar. 2008, doi: 10.1007/s11004-008-9156-6.

[7] "OpenCV," *opencv.org*. https://opencv.org.

[8] "Shape Matching using Hu Moments (C++ / Python) | Learn OpenCV," *Learn Open CV*, Dec. 11, 2018. https://learnopencv.com/shape-matching-using-hu-moments-c-python/#:~:text=Hu%20Moments%20(%20or%20rather%20Hu.
[9] J. Brownlee, "A Gentle Introduction to Information Entropy," *Machine Learning Mastery*, Oct. 13, 2019. https://machinelearningmastery.com/what-is-information-entropy/.

[10] R. C. Gonzalez and R. E. Woods, *Digital image processing*. New York, Ny: Pearson, 2018.

[11] "Leafsnap Dataset | Leafsnap: An Electronic Field Guide," *leafsnap.com*. http://leafsnap.com/dataset/ (accessed Jun. 25, 2021).

[12] "scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation," *scikit-learn.org*. https://scikit-learn.org/stable/index.html# (accessed Jun. 25, 2021).