

CMP9781M-BIG DATA ANALYTICS AND MODELLING.



UNIVERSITY OF LINCOLN

School of Computer Science

College of Science

University of Lincoln

Report submitted by Tracy Akowuah Acheampong
(28236126@students.lincoln.ac.uk)

in partial fulfilment of the requirements for the Degree of MSc (Hons) in
Data Science and Applied Analytics

May 2024

TABLE OF CONTENTS

Contents

ABSTRACT	3
1 INTRODUCTION	3
2 METHODOLOGY	4
2.1 Image Acquisition	4
2.2 Image Preprocessing	5
2.2.1 Image Loading and Conversion	5
2.2.2 Image Resizing	5
2.2.3 Grayscale Conversion	5
2.2.4 Image Segmentation	5
2.3 Feature Extraction	6
2.3.1 Contour Detection	6
2.3.2 Colour Histograms	7
2.4 Classification Process	9
2.4.1 Cross-Validation	9
2.4.2 Classifier Selection	10
2.4.3 Hyperparameter Tuning	13
2.4.4 Model Evaluation	13
3 RESULTS	14
3.1 Machine Learning Models	14
3.2 Deep Learning Models	15
3.3 Considerations for Future Work	16
4 CONCLUSION	16
5 REFERENCES	17

ABSTRACT

The most prominent feature that differentiates plants from each other is their leaves. From the broad, flat leaves of tropical rainforests to the needle-like foliage of coniferous forests, the variability of leaf morphology reflects adaptations to a wide range of environmental conditions. All parts of the leaf, including its shape, size, colour, texture and biochemical compositions provide more information than meets the eye.

This report explores the rich diversity of leaves by focusing on four distinct plant species: Mint, Guava, Tulsi, and Rose apple. By employing advanced image processing and machine learning techniques such as Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Fully Connected Neural Networks (FCNN), Convolutional Neural Networks (CNN) and Random Forest, I aim to classify leaves based on their morphological features. Through this classification process, we can discern characteristic traits unique to each plant species, contributing to our understanding of plant biodiversity and ecological dynamics. By uncovering patterns and trends in leaf morphology and anatomy, we gain insights into the evolutionary adaptations and ecological roles of different plants within their respective habitats. This approach not only aids in species identification but also provides valuable information for conservation efforts and ecosystem management.

1 INTRODUCTION

Leaves are the primary organs of photosynthesis in plants and exhibit immense diversity in structure, anatomy, and biochemical composition (Huntington, 2024). Their distinct characteristics reflect adaptations to various environmental conditions, making them crucial indicators of ecological processes and plant health (Shapiro, 2023). With over 350,000 species of vascular plants worldwide, accurate leaf classification is imperative for botanists, ecologists, and conservationists to understand plant biodiversity, ecosystem dynamics, and responses to environmental changes (Gardens, 2023). By identifying and categorizing leaf traits such as shape, size, venation patterns, and surface features, researchers can discern taxonomic relationships and infer functional traits associated with ecological strategies. For instance, leaves with specific adaptations such as drought tolerance or insect resistance can be identified and studied to explain underlying physiological mechanisms (Wu & Qian, 2021).

Automating leaf classification procedures by leveraging AI and Machine Learning methods enhances the efficiency and scalability of biodiversity assessments and ecological research. The algorithms trained on leaf images can rapidly classify specimens, enabling large-scale data collection and analysis. Such technology facilitates biodiversity monitoring, species conservation efforts, and ecosystem management in the face of global environmental changes and habitat loss (Hama, et al., 2024).

In this project, I aim to apply image processing and machine learning techniques to automate leaf classification for four distinct plant species: Mint, Guava, Tulsi, and Rose apple. These leaves also possess medicinal properties. For example, Tulsi (Fig 1) is used to treat insect bites and certain heart diseases (Garg, 2024), Guava (Fig 1) is a popular fruit used for different juices, its leaves are also used to treat stomach and intestinal pains as well as diabetes (WEB.MD, 2015), Rose apple leaves (Fig 1) help boost immunity and control diabetes (Wahome, 2022) while Mint(Fig 1) improves cold symptoms and helps manage stress (Boast, 2022). This further underscores the importance of effective image and object recognition processes in making life easier.

Through these advanced computational methods, I seek to contribute to interdisciplinary research endeavours and address pressing environmental challenges.



Figure 1 (Leaf classes)

2 METHODOLOGY

This section discusses the key stages of image processing that were leveraged to perform this task. These include Image Acquisition, Preprocessing, Feature extraction, and Image Classification (SimpliLearn, 2023), all of which are discussed below:

2.1 Image Acquisition

The Leaf Dataset used for this project was extracted from Kaggle.com. It contained 97 images of Mint, 52 images of Tulsi, 65 images of Guava, and 56 images of Rose apple. All essential libraries were imported to set up the environment.

```
[2]: import os
import numpy as np
import cv2
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.applications.resnet50 import preprocess_input
from sklearn.model_selection import StratifiedKFold, GridSearchCV
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

Figure 2

2.2 Image Preprocessing

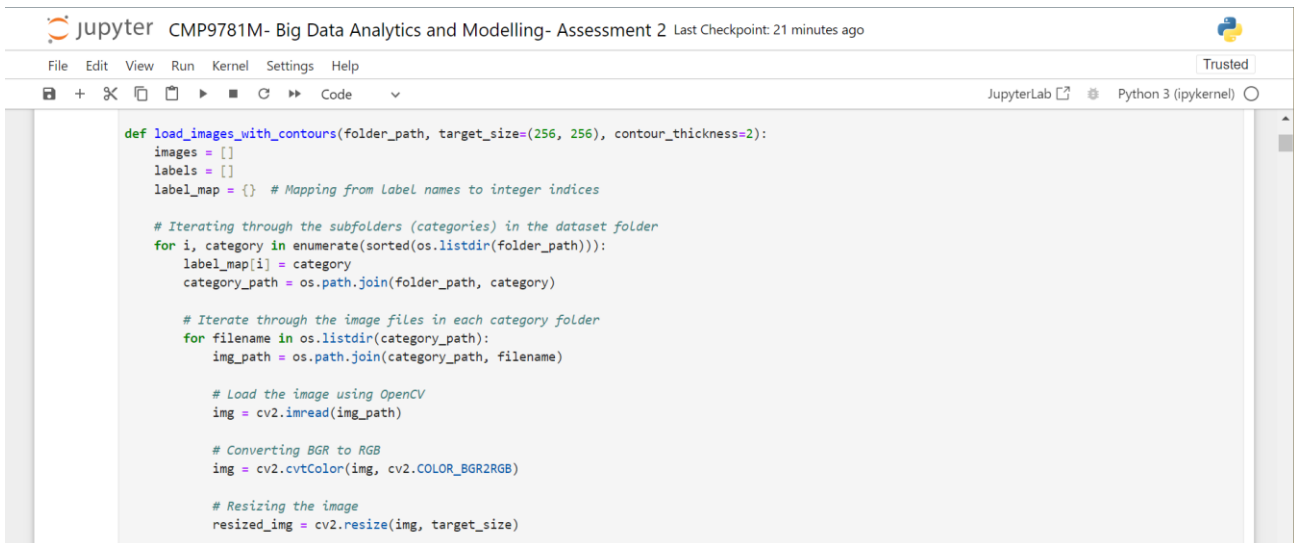
To begin any machine learning project, preprocessing is the crucial first step as it prepares, enhances, and transforms data into formats that are compatible with the algorithms (Patel, 2023). The preprocessing techniques used for this task are outlined below:

2.2.1 Image Loading and Conversion

Initially, the images were loaded using the OpenCV library and converted from their default BGR colour space to RGB to ensure consistency across different platforms and libraries (Fig 2).

2.2.2 Image Resizing

To ensure uniformity and reduce computational complexity, the images were resized to a standard size of (256, 256) pixels using the `cv2.resize()` function. This resizing step also helps in reducing memory requirements during feature extraction and classification.



```
def load_images_with_contours(folder_path, target_size=(256, 256), contour_thickness=2):
    images = []
    labels = []
    label_map = {} # Mapping from Label names to integer indices

    # Iterating through the subfolders (categories) in the dataset folder
    for i, category in enumerate(sorted(os.listdir(folder_path))):
        label_map[i] = category
        category_path = os.path.join(folder_path, category)

        # Iterate through the image files in each category folder
        for filename in os.listdir(category_path):
            img_path = os.path.join(category_path, filename)

            # Load the image using OpenCV
            img = cv2.imread(img_path)

            # Converting BGR to RGB
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            # Resizing the image
            resized_img = cv2.resize(img, target_size)
```

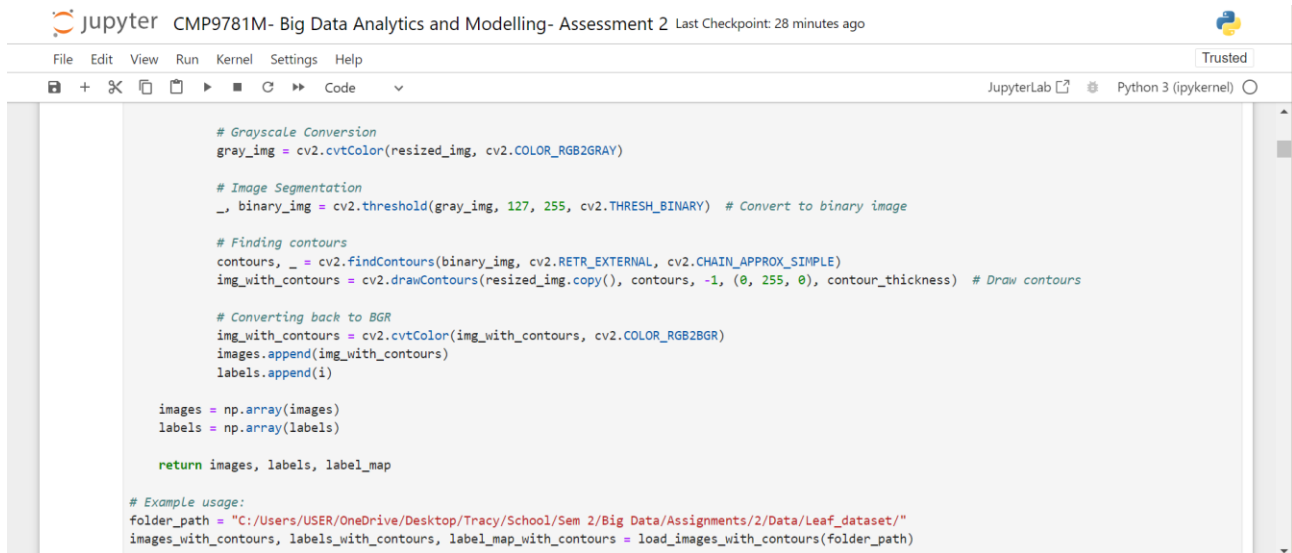
Figure 2

2.2.3 Grayscale Conversion

The resized images were converted to grayscale using the `cv2.cvtColor()` function (Fig 3). Grayscale conversion simplifies subsequent image-processing tasks and reduces computational overhead. This process involves the application of the formula: $Gray = R \times 0.2989 + G \times 0.5870 + B \times 0.1140$, which converts the images from their initial color spaces to a range of grayscale tones (Kuran, 2021).

2.2.4 Image Segmentation

Segmentation is the next step applied after grayscale conversion. It enhances contrast and highlights the features of interest. For this task, thresholding techniques were applied to convert the grayscale images into binary images. This step helps in segmenting the foreground i.e. the leaf from the background, thereby facilitating more accurate feature extraction (DataCarpentry, 2024).



```

# Grayscale Conversion
gray_img = cv2.cvtColor(resized_img, cv2.COLOR_RGB2GRAY)

# Image Segmentation
_, binary_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY) # Convert to binary image

# Finding contours
contours, _ = cv2.findContours(binary_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
img_with_contours = cv2.drawContours(resized_img.copy(), contours, -1, (0, 255, 0), contour_thickness) # Draw contours

# Converting back to BGR
img_with_contours = cv2.cvtColor(img_with_contours, cv2.COLOR_RGB2BGR)
images.append(img_with_contours)
labels.append(i)

images = np.array(images)
labels = np.array(labels)

return images, labels, label_map

# Example usage:
folder_path = "C:/Users/USER/OneDrive/Desktop/Tracy/School/Sem 2/Big Data/Assignments/2/Data/Leaf_dataset/"
images_with_contours, labels_with_contours, label_map_with_contours = load_images_with_contours(folder_path)

```

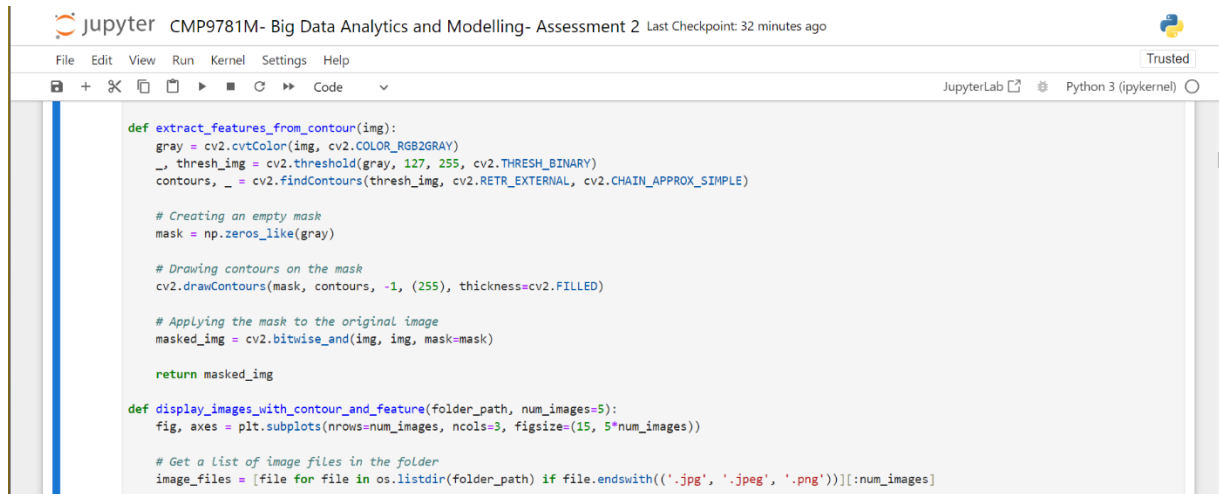
Figure 3

2.3 Feature Extraction

Feature extraction plays a crucial role in automated leaf classification as it involves extracting informative characteristics from the pre-processed images (nv5geospatialsoftware, 2024). In this section, we discuss the techniques used for feature extraction and their significance in capturing relevant information from the leaf images.

2.3.1 Contour Detection

Contours are fundamental features of leaf morphology and provide valuable information about leaf shape and structure. The `cv2.findContours()` function was utilized to detect contours in the binary images obtained after thresholding. These contours were then used to detect and extract various morphological features such as perimeter, area, and shape descriptors (LearnOpenCV, 2024). It was used with masking which is often used after contour detection to isolate the detected regions (objects) from the background. This allows for more focused analysis and feature extraction on the regions of interest without interference from the surrounding areas. (Cloudinary, 2024)



```

def extract_features_from_contour(img):
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    _, thresh_img = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresh_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Creating an empty mask
    mask = np.zeros_like(gray)

    # Drawing contours on the mask
    cv2.drawContours(mask, contours, -1, (255), thickness=cv2.FILLED)

    # Applying the mask to the original image
    masked_img = cv2.bitwise_and(img, img, mask=mask)

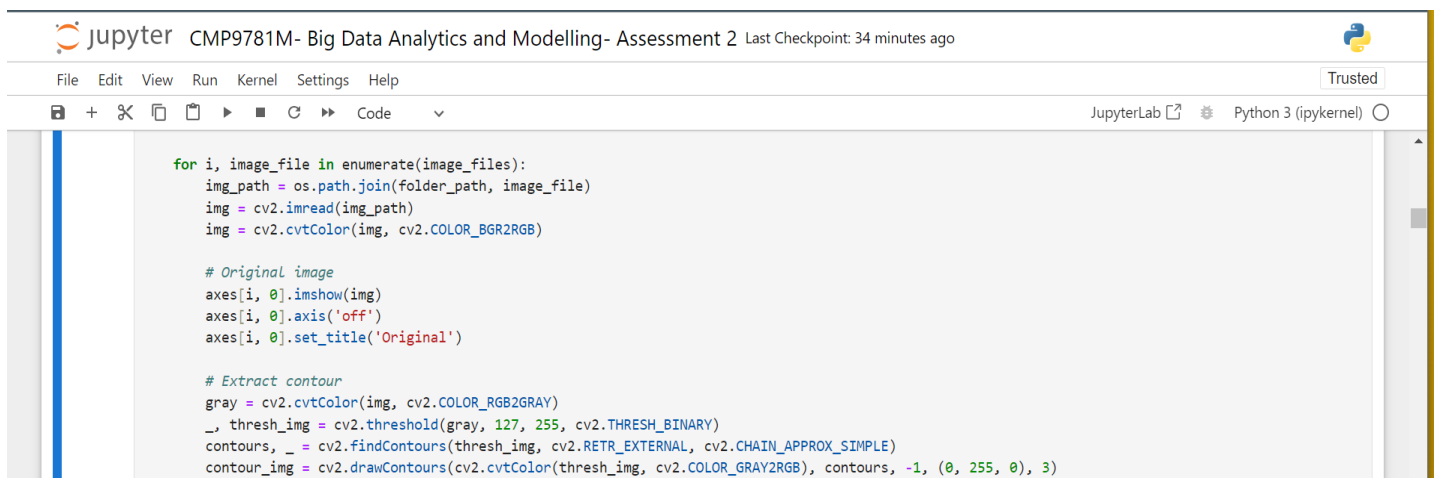
    return masked_img

def display_images_with_contour_and_feature(folder_path, num_images=5):
    fig, axes = plt.subplots(nrows=num_images, ncols=3, figsize=(15, 5*num_images))

    # Get a list of image files in the folder
    image_files = [file for file in os.listdir(folder_path) if file.endswith(('.jpg', '.jpeg', '.png'))][:num_images]

```

Figure 4



```

for i, image_file in enumerate(image_files):
    img_path = os.path.join(folder_path, image_file)
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Original image
    axes[i, 0].imshow(img)
    axes[i, 0].axis('off')
    axes[i, 0].set_title('Original')

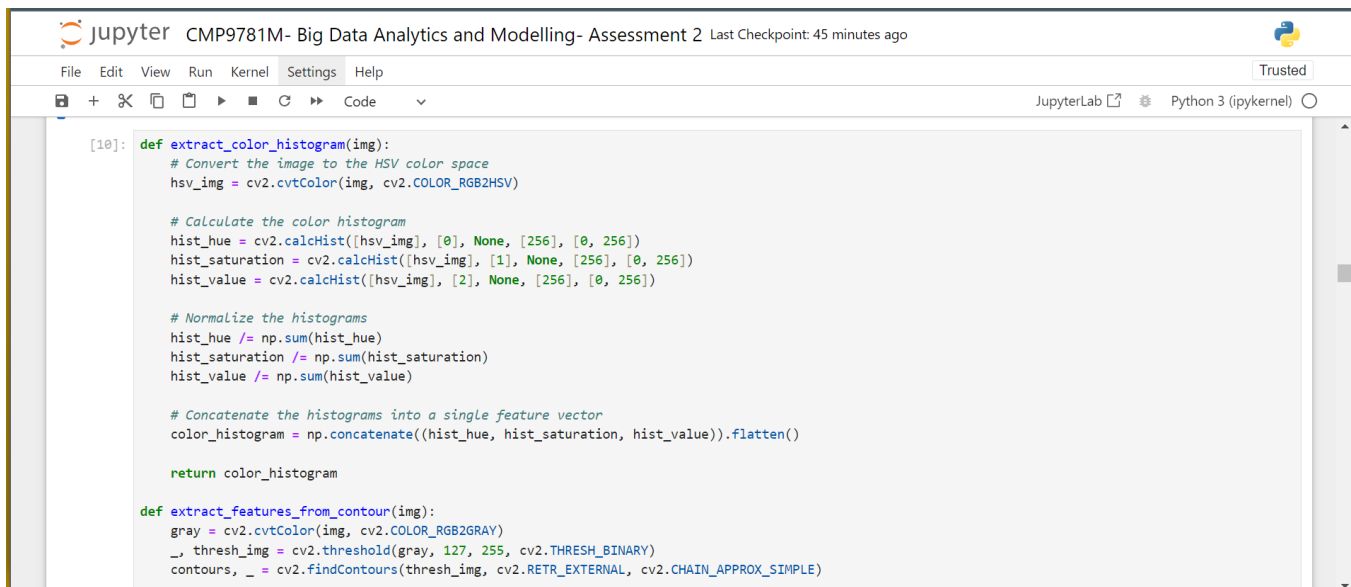
    # Extract contour
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    _, thresh_img = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresh_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contour_img = cv2.drawContours(cv2.cvtColor(thresh_img, cv2.COLOR_GRAY2RGB), contours, -1, (0, 255, 0), 3)

```

Figure 5

2.3.2 Colour Histograms

Colour information can also be informative in leaf classification tasks, especially for this task which had species exhibiting distinct colour variations. The `cv2.calcHist()` function was employed to compute colour histograms from the RGB images. These histograms capture the distribution of pixel intensities in different colour channels and provide valuable insights into leaf colour characteristics (Swain & Ballard, 2002).



```
[10]: def extract_color_histogram(img):
    # Convert the image to the HSV color space
    hsv_img = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)

    # Calculate the color histogram
    hist_hue = cv2.calcHist([hsv_img], [0], None, [256], [0, 256])
    hist_saturation = cv2.calcHist([hsv_img], [1], None, [256], [0, 256])
    hist_value = cv2.calcHist([hsv_img], [2], None, [256], [0, 256])

    # Normalize the histograms
    hist_hue /= np.sum(hist_hue)
    hist_saturation /= np.sum(hist_saturation)
    hist_value /= np.sum(hist_value)

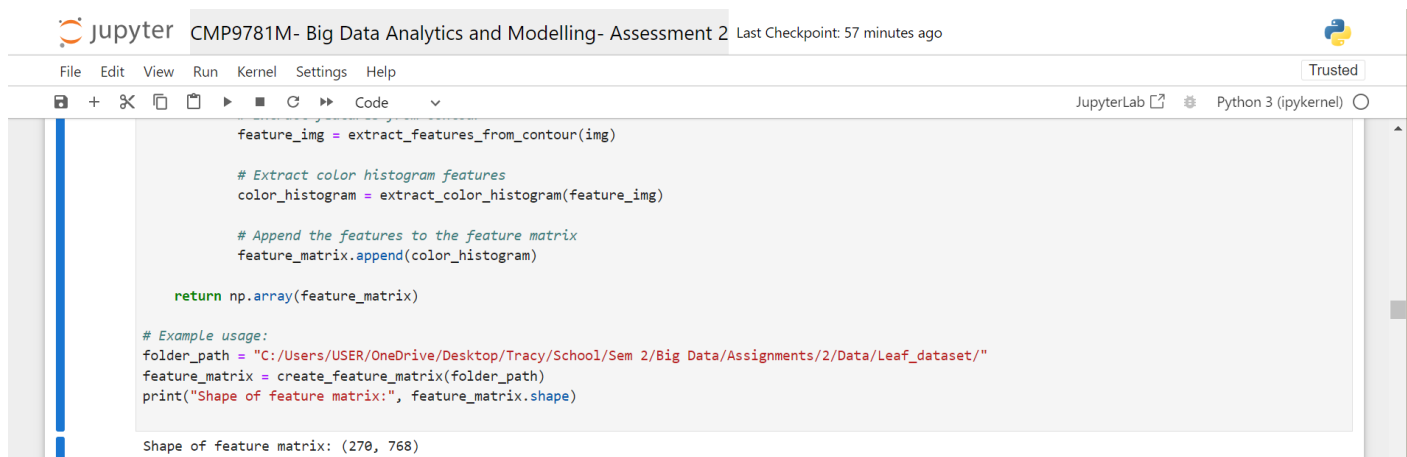
    # Concatenate the histograms into a single feature vector
    color_histogram = np.concatenate((hist_hue, hist_saturation, hist_value)).flatten()

    return color_histogram

def extract_features_from_contour(img):
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    _, thresh_img = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresh_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Figure 6

In the end, 768 features were extracted (Fig 7) and the visualisation of the processed image (Fig 8).



```
feature_img = extract_features_from_contour(img)

# Extract color histogram features
color_histogram = extract_color_histogram(feature_img)

# Append the features to the feature matrix
feature_matrix.append(color_histogram)

return np.array(feature_matrix)

# Example usage:
folder_path = "C:/Users/USER/OneDrive/Desktop/Tracy/School/Sem 2/Big Data/Assignments/2/Data/Leaf_dataset/"
feature_matrix = create_feature_matrix(folder_path)
print("Shape of feature matrix:", feature_matrix.shape)

Shape of feature matrix: (270, 768)
```

Figure 7

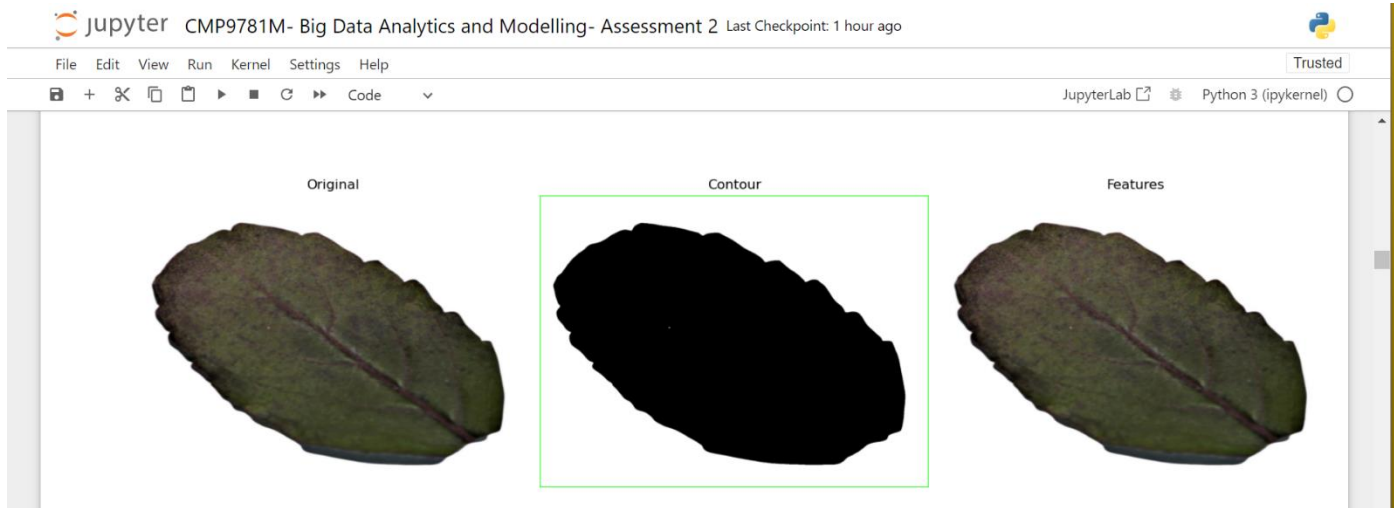


Figure 8

2.4 Classification Process

This section discusses the process of classifying the leaf images using 3 machine learning and 2 deep learning algorithms. The objective was to accurately assign each image to one of the predefined plant species categories: Mint, Guava, Tulsi, or Rose Apple.

2.4.1 Cross-Validation

To ensure the efficiency and reliability of the classification models, a technique called cross-validation was employed. Stratified K-Fold cross-validation with 10 folds was utilized. This approach ensures that each fold preserves the same proportion of each class as the original dataset, thereby reducing bias and variance in the evaluation process (Scikit-Learn, 2024).

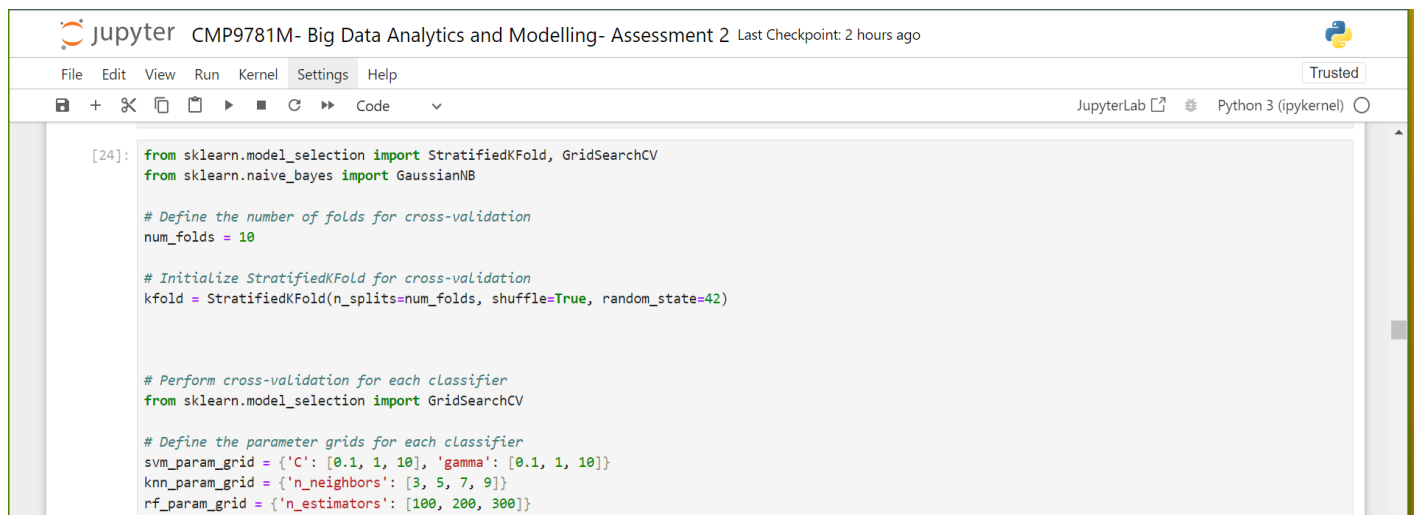


Figure 9

2.4.2 Classifier Selection

For this supervised learning task, three popular machine learning classifiers; Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Random Forest, Convolutional Neural Network (CNN), and Fully Convolutional Neural Network (FCNN) were chosen due to their versatility and effectiveness in handling classification tasks with both numerical and categorical data.

2.4.2.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful supervised learning algorithm primarily used for regression and classification tasks (Zarwal, 2023). It works by finding the optimal hyperplane that best separates the classes in the feature space and is particularly well-suited for tasks where the data is not linearly separable like this one. It is widely used in various domains, including image recognition, text classification, and bioinformatics and is known for its ability to handle high-dimensional data efficiently and its robustness to overfitting, making it suitable for a wide range of classification problems (IBM, 2021).

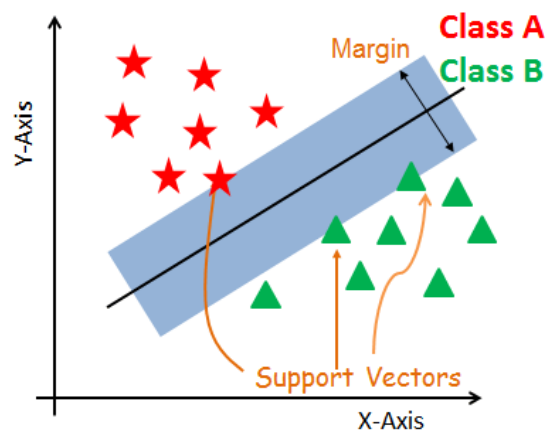


Figure 10 (How SVM works) (Navlani, 2019)

2.4.2.2 K-Nearest Neighbours (KNN)

K-Nearest Neighbours (KNN) was the second classifier. It is a simple and intuitive machine-learning algorithm used for classification and regression tasks. It operates on the principle of similarity, where the class of a new instance is determined by the majority class among its k nearest neighbours in the feature space (Sharma, 2023). It is suitable for both numerical and categorical data and is commonly used in pattern recognition, recommendation systems, and anomaly detection making it suitable for this task (IBM, 2024).

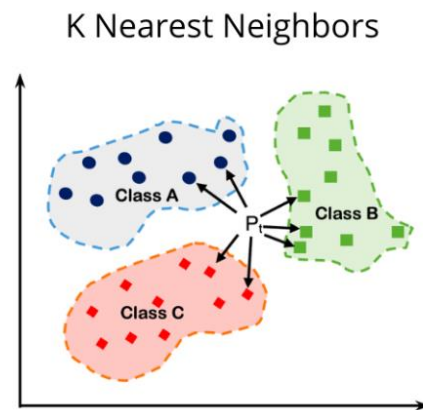


Figure 11 (KNN classifier) (Sachinsoni, 2023)

2.4.2.3 Random Forest

The only classifier that was not originally requested for use was Random Forest. It is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees (Scikit-Learn, 2024). It is known for its high accuracy, ability to withstand overfitting and to handle large datasets with high dimensionality and is mostly used in various applications, including image classification, gene expression analysis, and financial forecasting which is why it was chosen for this task. (Sruthi, 2024).

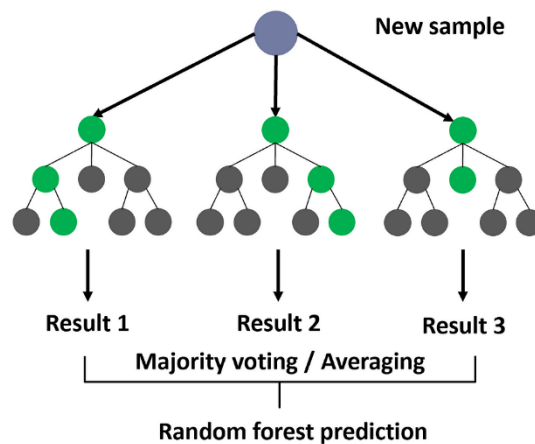


Figure 12(How Random Forest works) (Yehoshua, 2023)

2.4.2.4 Convolutional Neural Network (CNN)

The last two classifiers were deep learning models and CNN was one of them. It is a type of deep learning model commonly used for image classification, object detection, and image recognition tasks (Zakirizvi, 2024) and consists of multiple layers of convolutional and pooling operations followed by fully connected layers. CNNs automatically learn hierarchical features from raw pixel values, making them well-suited for image classification tasks where the spatial relationships between pixels are important which was essential for this leaf class (AnalyticsVidhya, 2024).

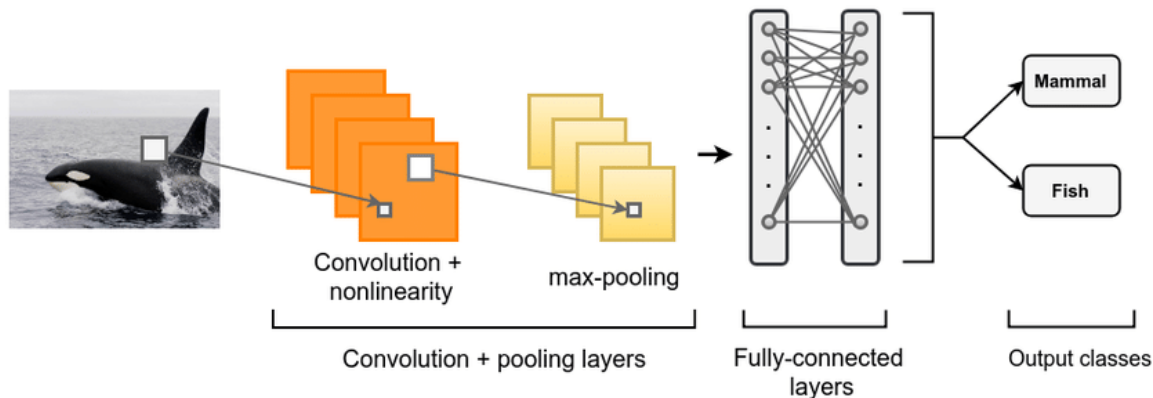


Figure 13 (CNN Pipeline)

2.4.2.5 Fully Connected Neural Network (FCNN)

FCNN, also known as a feedforward neural network, is a classic type of neural network where each neuron in one layer is connected to every neuron in the next layer (Ramsundar & Zadeh, 2017). The primary distinction between CNN AND FNN lies in their architecture: FCNNs (Fully Convolutional Neural Networks) are designed to be entirely convolutional, allowing their convolutional layers to accept inputs of different sizes and generate outputs of consistent size, CNNs on the contrary need consistent input and output dimensions. FCNNs are also more versatile and can be used for various machine-learning tasks, including classification (Singh, 2023).

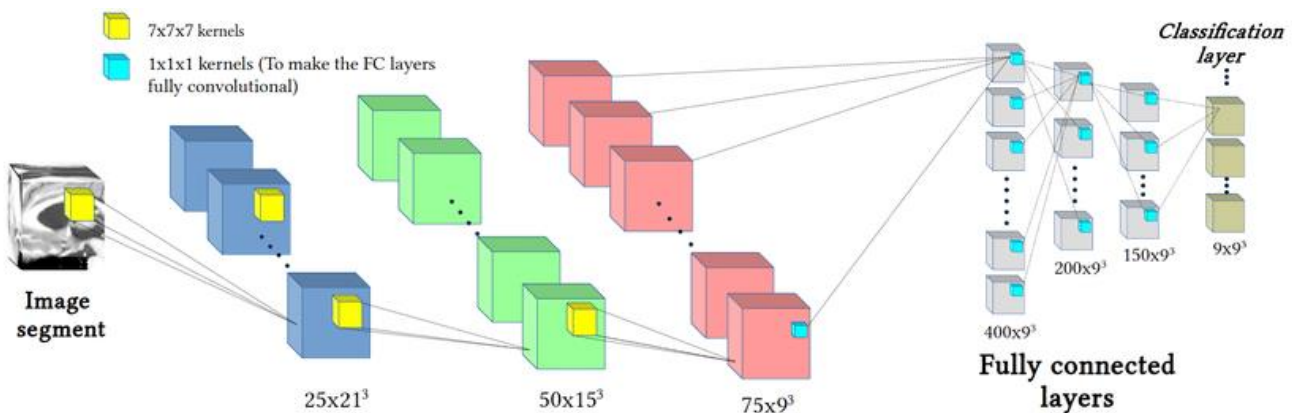
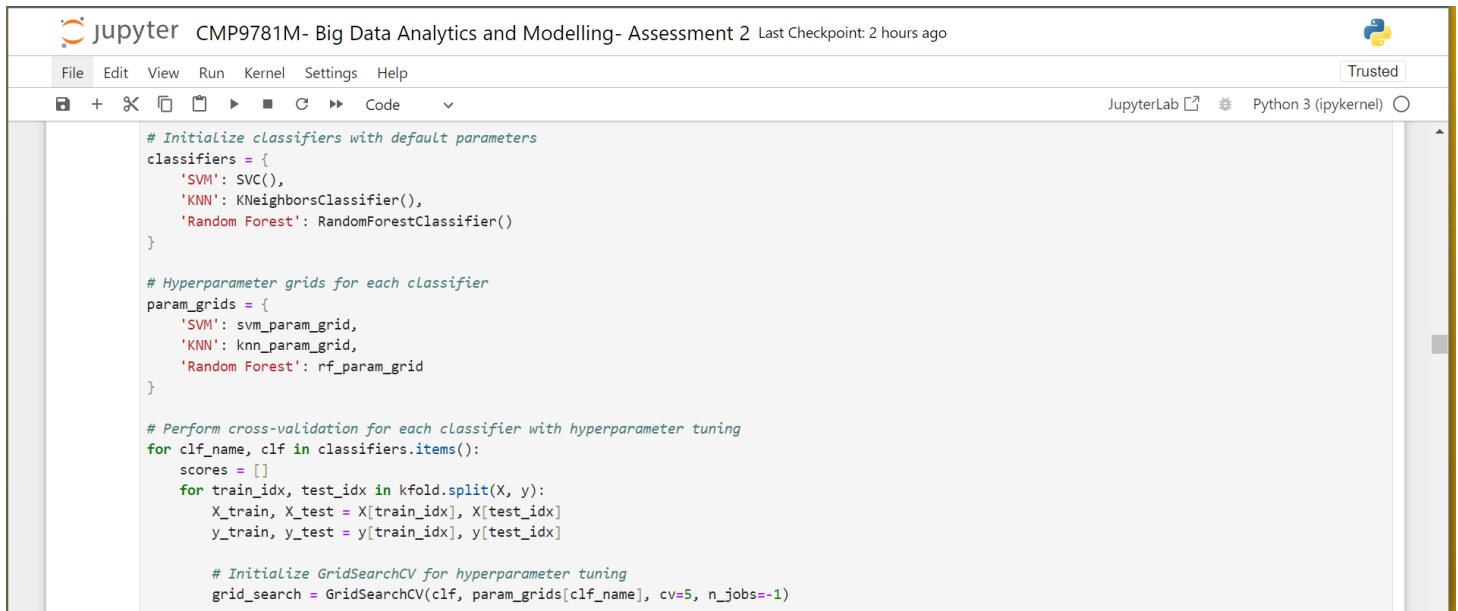


Figure 14 FCNN Pipeline (Dolz, 2017)

2.4.3 Hyperparameter Tuning

Before training the classifiers, hyperparameter tuning was performed using GridSearchCV. This technique systematically searches through a specified hyperparameter grid for each classifier to find the combination that maximizes model performance (Pandian, 2022). For SVM, the regularization parameter C and the kernel coefficient γ were tuned. For KNN, the number of neighbours ' k ' was adjusted and for Random Forest, the number of trees in the forest was optimized.



```

jupyter CMP9781M- Big Data Analytics and Modelling- Assessment 2 Last Checkpoint: 2 hours ago
File Edit View Run Kernel Settings Help Trusted
JupyterLab Python 3 (ipykernel)

# Initialize classifiers with default parameters
classifiers = {
    'SVM': SVC(),
    'KNN': KNeighborsClassifier(),
    'Random Forest': RandomForestClassifier()
}

# Hyperparameter grids for each classifier
param_grids = {
    'SVM': svm_param_grid,
    'KNN': knn_param_grid,
    'Random Forest': rf_param_grid
}

# Perform cross-validation for each classifier with hyperparameter tuning
for clf_name, clf in classifiers.items():
    scores = []
    for train_idx, test_idx in kfold.split(X, y):
        X_train, X_test = X[train_idx], X[test_idx]
        y_train, y_test = y[train_idx], y[test_idx]

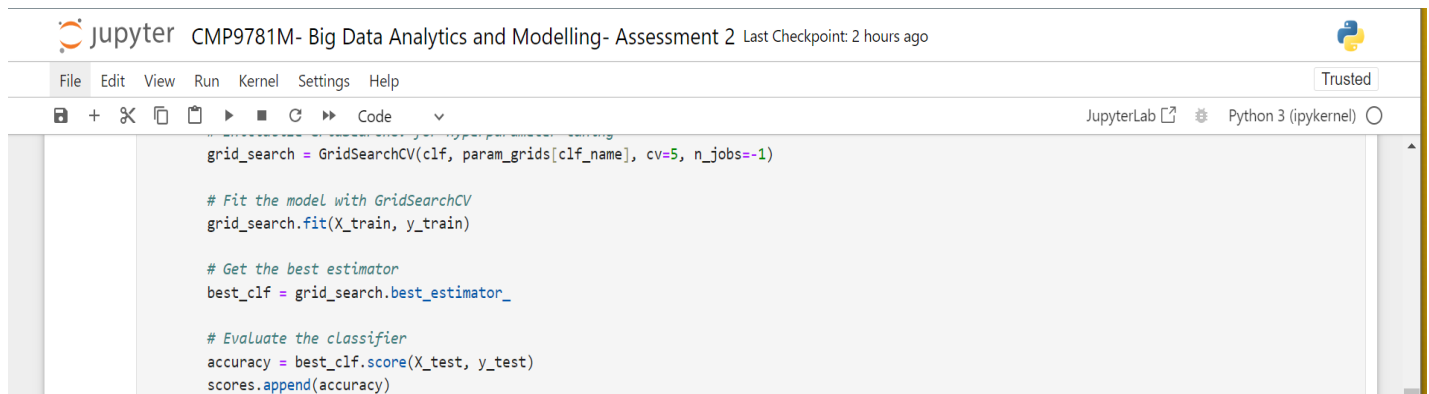
        # Initialize GridSearchCV for hyperparameter tuning
        grid_search = GridSearchCV(clf, param_grids[clf_name], cv=5, n_jobs=-1)

```

Figure 15

2.4.4 Model Evaluation

The performance of each classifier was evaluated using the mean accuracy metric calculated over all cross-validation folds. This metric represents the average proportion of correctly classified instances across different iterations of the cross-validation process. By assessing the mean accuracy of each classifier, we can determine which model performs best on classifying the leaf classification.



```

jupyter CMP9781M- Big Data Analytics and Modelling- Assessment 2 Last Checkpoint: 2 hours ago
File Edit View Run Kernel Settings Help Trusted
JupyterLab Python 3 (ipykernel)

        grid_search = GridSearchCV(clf, param_grids[clf_name], cv=5, n_jobs=-1)

        # Fit the model with GridSearchCV
        grid_search.fit(X_train, y_train)

        # Get the best estimator
        best_clf = grid_search.best_estimator_

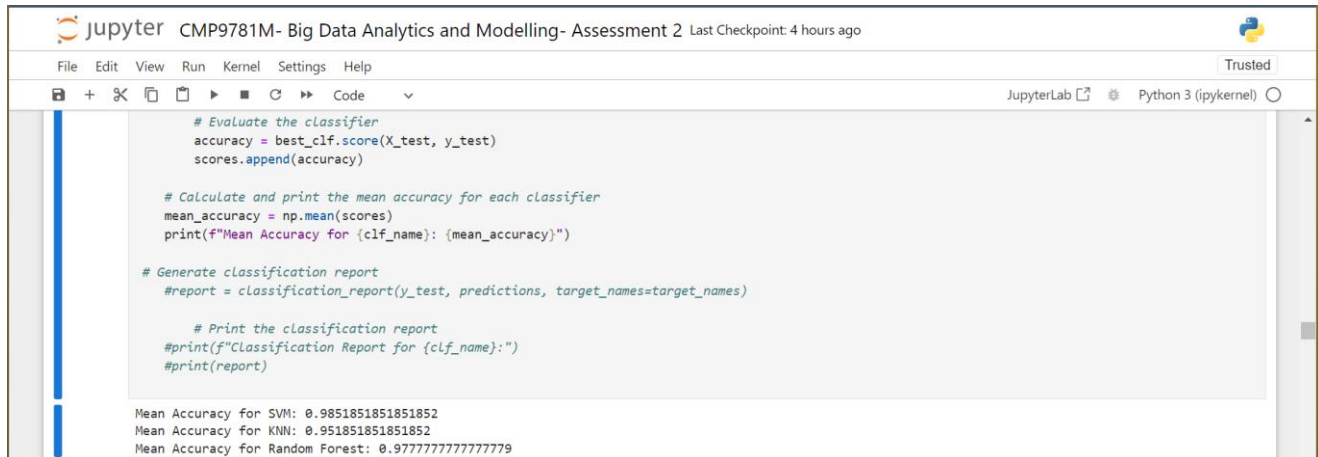
        # Evaluate the classifier
        accuracy = best_clf.score(X_test, y_test)
        scores.append(accuracy)

```

3 RESULTS

Since the classifiers were all used on the same data, their competence was measured by their ability to accurately classify the leaf datasets. As stated earlier, three machine learning classifiers were used against two deep learning models. The overall evaluation would then be carried out in two folds. Firstly by comparing the overall performance of machine learning models against deep learning models and vice versa and then finally on the overall best model.

3.1 Machine Learning Models



The screenshot shows a JupyterLab window titled 'CMP9781M- Big Data Analytics and Modelling- Assessment 2' with a 'Last Checkpoint: 4 hours ago' status. The interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations and code execution. The main area contains a code editor with the following Python code:

```
# Evaluate the classifier
accuracy = best_clf.score(X_test, y_test)
scores.append(accuracy)

# Calculate and print the mean accuracy for each classifier
mean_accuracy = np.mean(scores)
print(f"Mean Accuracy for {clf_name}: {mean_accuracy}")

# Generate classification report
#report = classification_report(y_test, predictions, target_names=target_names)

# Print the classification report
#print(f"Classification Report for {clf_name}:")
#print(report)
```

Below the code editor, the output of the code is displayed:

```
Mean Accuracy for SVM: 0.9851851851851852
Mean Accuracy for KNN: 0.951851851851852
Mean Accuracy for Random Forest: 0.9777777777777779
```

Figure 17

The image above displays the result after training the model and these are interpreted below:

The Support Vector Machine (SVM) model achieved an impressive mean accuracy of approximately 98.52%, indicating its effectiveness in classifying the data. With its ability to find the optimal hyperplane that best separates the classes in high-dimensional space, SVM demonstrates too good a performance in a variety of classification tasks. Results that high raise speculations of overfitting.

The K-Nearest Neighbours (KNN) algorithm, on the other hand, achieved a mean accuracy of approximately 95.19%. KNN is a simple yet powerful non-parametric algorithm that relies on the similarity of instances to classify new data points. While it may not have achieved the highest accuracy among the models tested, its performance is still notable, especially considering its simplicity and ease of implementation. It is also slightly too high and might have had instances of overfitting.

Similar to SVM, the Random Forest classifier attained a mean accuracy of approximately 97.78%. As an ensemble learning method, it is assumed that has the ability to handle high-dimensional data and mitigate overfitting which is a reason why it was chosen. Again, 97% suggests that the model is either very good at identifying and classifying the leaves or that there was overfitting.

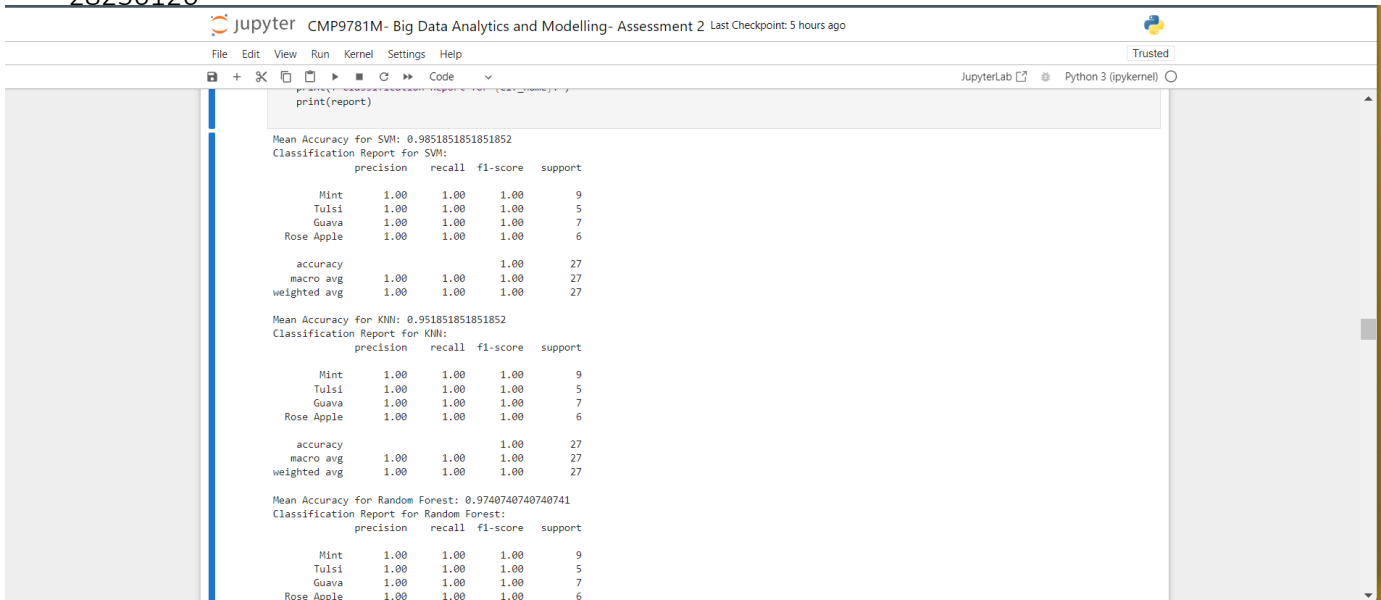


Figure 18

The classification report in Figure 18 further supports the assumption of overfitting, seeing as how all the scores for precision, recall, and f1 scores are 100% for the three models. Overfitting is a problematic phenomenon in machine learning, where the model performs well on the training data but fails to generalize effectively to unseen data (AWS, 2024). That aside, SVM appears to be the best model in the group.

3.2 Deep Learning Models

For the deep learning models Convolutional Neural Network (CNN) achieved an accuracy score of 88.9%, indicating that it correctly classified approximately 88.9% of the leaf images in the dataset. On the other hand, the Fully Connected Neural Network (FCNN) performed slightly better with an accuracy score of 94.4%, suggesting that it accurately classified approximately 94.4% of the leaf images.

These results demonstrate that both CNN and FCNN are effective in classifying the leaf images, with the FCNN model showing a slightly higher accuracy compared to the CNN model. Compared to the Machine Learning models (Figure 19), these accuracy scores suggest very little to no overfitting. However, several other factors were considered before selecting these models including computational efficiency and model complexity.

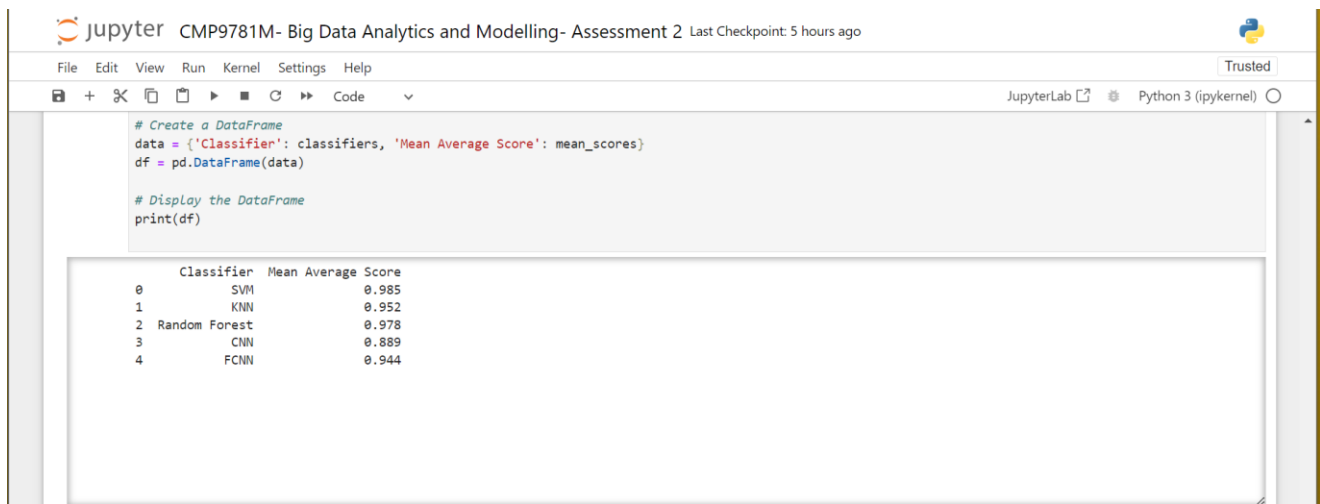


Figure 19

3.3 Considerations for Future Work

While these classification models have shown promising results, there may be opportunities for further optimization and refinement. To avoid or reduce the occurrence of overfitting in future work more feature extraction techniques could be explored, different hyperparameter settings could be experimented with, or more advanced machine learning algorithms could be incorporated to enhance the classification accuracy even further. This does not mean that a model being able to accurately classify all instances is wrong or impossible, these are just precautionary measures needed to ensure data integrity.

4 CONCLUSION

In conclusion, this task explored the application of machine learning and deep learning techniques for leaf classification, focusing on four distinct plant species: Mint, Guava, Tulsi, and Rose Apple. By leveraging various classifiers including Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest, Convolutional Neural Network (CNN), and Fully Connected Neural Network (FCNN), we aimed to automate the process of leaf classification and enhance our understanding of plant biodiversity.

The results revealed high accuracy scores across all models, with SVM, KNN, and Random Forest achieving mean accuracy scores of approximately 98.5%, 95.2%, and 97.8% respectively. Furthermore, deep learning models such as CNN and FCNN demonstrated promising performance, with accuracy scores of 88.9% and 94.4% respectively.

These findings underscore the effectiveness of machine learning and deep learning approaches in accurately classifying leaf images, thus facilitating biodiversity monitoring, species conservation efforts, and ecological research. Moreover, the study highlights the importance of selecting appropriate classifiers based on factors such as model complexity, computational efficiency, and interpretability.

Moving forward, further research will be required to optimise model hyperparameters, explore ensemble learning techniques, and incorporate domain-specific features to improve classification accuracy and robustness. Additionally, efforts to integrate advanced image processing methods and domain knowledge can enhance the

interpretability of models and facilitate the discovery of novel insights into plant biodiversity and ecosystem dynamics. Overall, the application of machine learning and deep learning in leaf classification holds great potential for advancing ecological research, environmental conservation, and sustainable development initiatives.

5 REFERENCES

Isman, M. . B., 2019. Challenges of Pest Management in the Twenty First Century: New Tools and Strategies to Combat Old and New Foes Alike. *Frontiers in Agronomy*, Volume 1.

Li , Y. et al., 2020. Computers and Electronics in Agriculture. *Science Direct*, Volume 169.

AnalyticsVidhya, 2024. *Analytics Vidhya*. [Online]

Available at: https://courses.analyticsvidhya.com/courses/convolutional-neural-networks-cnn-from-scratch?utm_source=blog&utm_medium=learn-image-classification-cnn-convolutional-neural-networks-3-datasets#:~:text=A%20Convolutional%20Neural%20Network%20is%20a%20powerful%20model%20for%20image%20classification.,text=Convolutional%20Neural%20Networks%20are%20a%20type%20of%20deep%20learning%20model%20that%20can%20be%20used%20to%20classify%20images%20into%20different%20categories%20based%20on%20the%20features%20extracted%20from%20the%20input%20image.
[Accessed 7 May 2024].

Armi, L. & Fekri-Ershad, S., 2019. Texture image analysis and texture classification methods - A review. *Cornell University*, 13 April.

AWS, 2024. AWS. [Online]

Available at: <https://aws.amazon.com/what-is/overfitting/#:~:text=Overfitting%20is%20an%20undesirable%20machine%20learning%20behavior%20that%20occurs%20when%20the%20machine%20learning%20model%20gives%20accurate%20predictions%20for%20Otraining%20data%20but%20not%20for%20>
[Accessed 8 May 2024].

Boast, G., 2022. HEALTH BENEFITS OF MINT. *Equinox Kombucha*, 14 September.

Chotai, N., 2020 . *A Kilo of Spices*. [Online]

Available at: <https://www.akospices.com/blogs/news/an-informative-guide-about-cooking-with-tulsi-or-tulsi-leaf-powder>
[Accessed 3 May 2024].

Clouinary, 2024. *Image Masking*. [Online]

Available at: <https://cloudinary.com/glossary/image-masking#:~:text=Image%20masking%20is%20a%20technique%20used%20in%20photo%20editing%20to%20separate%20or%20isolate%20specific%20areas%20of%20an%20image%20from%20the%20rest%2C> [Accessed 5 May 2024].

DataCarpentry, 2024. Thresholding. *Data Carpentry*, 12 March.

Datta, A. K., Chowdhury, S. & Maity, S., 2012. Jute Biology, Diversity, Cultivation, Pest Control, Fiber Production and Genetics. *Research Gate*, April.

Dolz, J., 2017. *Research Gate*. [Online]

Available at: <https://www.researchgate.net/profile/Jose-Dolz-2/publication/316244066/figure/fig1/AS:484863246639105@1492611814638/The-baseline-FCNN-architecture-CNN-base-composed-of-3-convolutional-layers-with.png>
[Accessed 5 May 2024].

Gardens, K. R. B., 2023. *State of the World's Plants and Fungi*, s.l.: Kew Royal Botanic Gardens.

Garg, D. P., 2024. Tulsi (Holy Basil): Health Benefits, Uses and Nutritional Value. *Pharm Easy*, 3 May.

Gómez-Camperos, J. . A., Jaramillo , H. . Y. & Guerrero-Gómez , G., 2022. Digital image processing techniques for detection of pests and diseases in crops: a review. *Universidad del Valle*, 24(1).

Hama, H. M., Abdulsamad , T. S. & Omer, S. M., 2024. Houseplant leaf classification system based on deep learning algorithms. *Journal of Electrical Systems and Information Technology*, Volume 18.

Huntington, T., 2024. *Survival through Adaptation: Plant Parts & Patterns*. [Online]

Available at: <https://huntington.org/educators/learning-resources/survival-through-adaptation/plant-parts-patterns/leaves#:~:text=Leaves%20are%20one%20of%20the%20three%20organs%20of%20a%20plant.%20The%20most%20important%20job%20of%20a%20leaf%20is%20to%20make%20food%20of>
[Accessed 1 May 2024].

IBM, 2021. *IBM*. [Online]

Available at: <https://www.ibm.com/docs/en/spss-modeler/saas?topic=models-how-svm-works#:~:text=this%20topic%20helpful%3F-,How%20SVM%20Works,-Last%20Updated%3A%202021>
[Accessed 5 May 2024].

IBM, 2024. *IBM*. [Online]

Available at:
<https://www.ibm.com/topics/knn#:~:text=It%20is%20commonly%20used%20for%20simple%20recommendation%20systems%2C%20pattern%20recognition%2C%20data%20mining%2C%20financial%20market%20predictions%2C%20intrusion%20detection%2C%20and%20more.%C2%A0>
[Accessed 5 May 2014].

Kuran, E. C., 2021. A Guide to Contrast Enhancement: Transformation Functions, Histogram Sliding, Contrast Stretching and Histogram Equalization Methods with Implementations from Scratch using OpenCV Python. *Medium*, 28 March.

LearnOpenCV, 2024. *LearnOpenCV.com*. [Online]

Available at: <https://learnopencv.com/contour-detection-using-opencv-python-c/#:~:text=Using%20contour%20detection%2C%20we%20can,image%20segmentation%2C%20detection%20and%20recognition.>
[Accessed 3 May 2024].

Maari, T., 2011. *Wikipedia*. [Online]

Available at:
https://upload.wikimedia.org/wikipedia/commons/2/2a/Jute_Rope_%28%E0%AE%9A%E0%AE%A3%E0%AE%18

B2%E0%AF%8D %E0%AE%95%E0%AE%AF%E0%AE%BF%E0%AE%B1%E0%AF%81%29.jpg
[Accessed 29 April 2024].

Navlani, A., 2019. *Data Camp*. [Online]

Available at:

http://res.cloudinary.com/dyd911kmh/image/upload/f_auto,q_auto:best/v1526288453/index3_souoaz.png
[Accessed 5 May 2024].

nv5geospatialsoftware, 2024. *nv5geospatialsoftware.com*. [Online]

Available at:

[https://www.nv5geospatialsoftware.com/docs/FXExampleBasedTutorial.html#:~:text=Feature%20Extraction%20uses%20an%20object%2Dbased%20approach%20to%20classify%20imagery%2C%20where%20an%20object%20\(also%20called%20segment\)%20is%20a%20group%20of%20pixels%20wit](https://www.nv5geospatialsoftware.com/docs/FXExampleBasedTutorial.html#:~:text=Feature%20Extraction%20uses%20an%20object%2Dbased%20approach%20to%20classify%20imagery%2C%20where%20an%20object%20(also%20called%20segment)%20is%20a%20group%20of%20pixels%20wit)
[Accessed 3 May 2024].

Pandian, S., 2022. A Comprehensive Guide on Hyperparameter Tuning and its Techniques. *Analytics Vidhya*, 20 October.

Patel, M., 2023. The Complete Guide to Image Preprocessing Techniques in Python. *Medium*, 23 October.

Ramsundar, B. & Zadeh, R. B., 2017. *TensorFlow for Deep Learning*. First Edition ed. Sebastopol: O'Reilly Media, Inc..

Sachinsoni, 2023. *Medium*. [Online]

Available at: https://miro.medium.com/v2/resize:fit:750/format:webp/0*2_qzcm2gSe9l67al.png
[Accessed 3 May 2024].

Scikit-Learn, 2024. *Scikit-Learn*. [Online]

Available at: [https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)

[learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)
[Accessed 5 May 2024].

Scikit-Learn, 2024. *Scikit-Learn.org*. [Online]

Available at: [https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#:~:text=This%20cross%2Dvalidation%20object%20is%20a%20variation%20of%20KFold%20that%20returns%20stratified%20folds.%20The%20folds%20are%20made%20by%20preserving)

[learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#:~:text=This%20cross%2Dvalidation%20object%20is%20a%20variation%20of%20KFold%20that%20returns%20stratified%20folds.%20The%20folds%20are%20made%20by%20preserving](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#:~:text=This%20cross%2Dvalidation%20object%20is%20a%20variation%20of%20KFold%20that%20returns%20stratified%20folds.%20The%20folds%20are%20made%20by%20preserving)
[Accessed 7 May 2024].

Shapiro, L., 2023. Why do trees have differently shaped leaves?. *The Washington Post*, 16 November.

Sharma, S., 2023. K-Nearest Neighbour: The Distance-Based Machine Learning Algorithm. *Analytics Vidhya*, 9 November.

SimpliLearn, 2023. What Is Image Processing : Overview, Applications, Benefits, and More. *SimpliLearn*, 11 October.

Singh, D., 2023. *Quora*. [Online]

Available at: [https://www.quora.com/How-is-Fully-Convolutional-Network-FCN-different-from-the-original-Convolutional-Neural-Network-CNN#:~:text=Fully%20Convolutional%20Networks%20\(FCN\)%20are%20an%20improved%20version%20of%20Convolutional%20Neural%20Networks%20\(CNN\).%20](https://www.quora.com/How-is-Fully-Convolutional-Network-FCN-different-from-the-original-Convolutional-Neural-Network-CNN#:~:text=Fully%20Convolutional%20Networks%20(FCN)%20are%20an%20improved%20version%20of%20Convolutional%20Neural%20Networks%20(CNN).%20)

[Accessed 5 May 2024].

Sruthi, R. E., 2024. Understand Random Forest Algorithms With Examples (Updated 2024). *Analytics Vidhya*, 19 April.

Swain, M. . J. & Ballard, D. H., 2002. Color Histogram. *Science Direct*.

Wahome, C., 2022. *WebMD.com*. [Online]

Available at: <https://www.webmd.com/diet/health-benefits-rose-apples#:~:text=Rose%20apples%20have%20a%20variety%20of%20health%20benefits%2C%20like%20improving%20heart%20health%2C%20promoting%20immunity%2C%20controlling%20diabetes>

[Accessed 3 May 2024].

WEB.MD, 2015. *WEB MD*. [Online]

Available at: <https://www.webmd.com/vitamins/ai/ingredientmono-1133/guava#:~:text=People%20use%20guava%20leaf%20for%20stomach%20and%20intestinal%20conditions%2C%20pain%2C%20diabetes%2C%20and%20wound>

[Accessed 1 May 2024].

Wikipedia, 2021. *Jute*. [Online]

Available at: [https://en.wikipedia.org/wiki/File:Jute_Field_Bangladesh_\(7749587518\).jpg](https://en.wikipedia.org/wiki/File:Jute_Field_Bangladesh_(7749587518).jpg)

[Accessed 29 April 2024].

Wu, P. & Qian, Z., 2021. Leaf Classification Based on Convolutional Neural Network. *Journal of Physics: Conference Series*.

Yehoshua, D. R., 2023. *Medium*. [Online]

Available at: https://miro.medium.com/v2/resize:fit:1100/format:webp/1*jE1Cb1Dc_p9WEOPMkC95WQ.png

[Accessed 5 May 2024].

Zakirizvi, M., 2024 . Image Classification Using CNN (Convolutional Neural Networks). *Analytics Vidhya* , 02 May.

Zarwal, A., 2023. Image classification using Support Vector Machine (SVM) in Python. *Geeks4Geeks*, 21 March.

