Journal of Theoretical and Applied Information Technology
15th August 2023. Vol.101. No 15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COLLABORATIVE ANT COLONY OPTIMIZATION-ASSISTED SUPPORT VECTOR MACHINE FOR ACCURATE COTTON LEAF DISEASE CLASSIFICATION AND YIELD PREDICTION

S.GOVINDASAMY[1] , D.JAYARAJ[2]

[1]Research Scholar, Department of Computer and Information Science, Annamalai University,

[2] Assistant Professor/ Programmer, Department of Computer Science and Engineering,

Annamalai University, Tamilnadu, India

E-mail: [1] govindasamy1412@gmail.com, [2] jayarajvnr@gmail.com

ABSTRACT

Cotton leaf diseases pose a significant threat to the economic viability of cotton farming, leading to substantial yield losses. However, the lack of a reliable classification system and accurate yield prediction based on disease profiles hinders effective disease management and resource allocation. This research tackles these issues by recommending a novel method for predicting cotton production and classifying cotton leaf diseases using a combination of Ant Colony Optimization (ACO) and Support Vector Machines (SVM). The research objectives include developing a customized Collaborative ACO (CACO) algorithm for feature selection, implementing an SVM model for disease classification, and integrating the CACO-SVM framework for accurate disease identification and yield prediction. The suggested method will be tested on a labelled dataset of cotton leaf samples annotated with disease information. The importance of this study rests in its ability to allocate resources to cotton growing better and improve disease management tactics. Farmers can adopt targeted management practices, reduce production costs, and mitigate the environmental impact of generalized treatment approaches by accurately classifying cotton leaf diseases and predicting crop yields. The outcomes of this study are expected to contribute to improved agricultural practices, increased profitability for cotton farmers, and enhanced sustainability in cotton production. The proposed CACO-SVM framework has the potential to revolutionize disease identification and yield prediction in cotton farming, empowering farmers to make informed decisions and achieve higher crop management efficiency.

Keywords: *Classification, Ant Colony Optimization, Support Vector Machines, Yield Prediction, And Cotton Leaf Disease*

1. INTRODUCTION

This Leaf disease identification is crucial in plant health monitoring and disease management. In this process, a series of steps are followed to identify and classify diseases affecting plant leaves accurately. The first step involves data collection, gathering a dataset comprising healthy and diseased leaf images. Preprocessing techniques are then applied to enhance and normalize the images. The images may undergo optional segmentation to isolate the leaf region. Features are extracted from the segmented or preprocessed images, and a subset of relevant features may be selected. A suitable model is chosen, and its hyperparameters are tuned for optimal performance. The model is trained using the training set, evaluated using the validation set, and the identified diseases are reported as the output result. The general steps involved in lead disease identification are given in Figure 1.

Cotton is a major cash crop grown worldwide and one of the most important fibre crops in the textile industry. Cotton fibres are used to manufacture clothing, towels, bed sheets, and other textiles. Cotton plants are also a source of cottonseed oil, used for cooking and producing margarine, soap, and cosmetics [1]. Despite its economic importance, cotton production faces numerous challenges, including pests and diseases that can reduce yields and quality. Cotton Leaf Diseases (CLDs) are a significant concern among these diseases. CLDs can be caused by bacteria,

viruses, or fungi and can cause symptoms such as leaf spots, wilting, discolouration, and defoliation. Some of the common CLDs include cotton leaf curl virus (CLCuV), Fusarium wilt, and bacterial blight. CLDs can severely impact cotton yields, significantly reducing the number and quality of cotton fibres produced. In addition, to yield losses, CLDs can also cause the cotton plant to weaken, making it more susceptible to other diseases and pests[2]. Traditionally, CLDs have been identified through manual classification based on visual observations of symptoms. However, this process can be time-consuming and prone to errors, as different diseases can have similar symptoms. Manual classification's lack of accuracy and speed can lead to disease management and control delays [3].

Manual classification based on visual observation is the traditional method used to identify and classify Cotton Leaf Diseases (CLDs). In this method, the plant pathologist visually inspects the cotton plants, examines the leaves, and identifies the symptoms of the diseases [4]. However, manual classification has several challenges associated with it. Firstly, it is a time-consuming process that requires a trained plant pathologist to identify and classify the diseases accurately. This process can be difficult and time-consuming, especially for large cotton farms or plantations with thousands of cotton plants [5]. The second challenge is the subjectivity of visual observations. Different plant pathologists may have varying interpretations of the symptoms of the diseases, leading to inconsistencies in disease classification. This subjectivity can lead to misdiagnosis and ineffective treatments or management strategies. The manual classification may not be efficient in identifying and diagnosing CLDs at an early stage [6]. CLDs can spread quickly, and early detection is crucial to prevent their spread and reduce their impact on crop yields. In addition, the manual classification may not be able to differentiate between similar diseases, leading to confusion and misdiagnosis. As a result of these limitations, there is a need for more efficient and accurate methods for identifying and classifying CLDs. This has led to exploring machine learning (ML) as a potential solution. With ML, images of infected cotton leaves can be analyzed, and the disease can be identified and classified automatically [7].

To overcome these limitations, machine learning (ML) advancements have provided new opportunities for automating CLD detection and classification. Machine learning involves using computer algorithms that learn from data and improve their performance over time[8]. Using computer vision and ML algorithms, researchers can train machines to recognize and classify CLDs from images of infected leaves. ML algorithms can analyze many images quickly and accurately, allowing for real-time detection and classification of CLDs. This can help farmers and researchers identify and manage CLDs early, preventing their spread and limiting their impact on crop production [9]. ML in CLD detection and classification offers several advantages over traditional methods. It is faster, more accurate, and can be performed on a larger scale. ML algorithms can also learn from new data, improving disease detection and classification. Overall, the use of ML in CLD detection and classification has the potential to revolutionize cotton disease management and enhance cotton yields. Because of the expanding availability of digital tools and the importance of environmentally responsible farming, ML-based approaches will likely become more prevalent in the coming years [10].

Bio-inspired optimization [11],[12, 13, 22–25, 14–21] has emerged as a promising approach, harnessing natural principles to tackle diverse research problems. By mimicking biological systems, these algorithms exhibit adaptability, robustness, and scalability, making them applicable across engineering, data mining, and image processing domains. With ongoing advancements, bio-inspired optimization unlocks new possibilities for efficient and effective problem-solving.

1.1. Motivation

Cotton is a major cash crop that significantly impacts the worldwide textile industry & agricultural economy. However, cotton leaf diseases can cause substantial yield losses, affecting farmers' incomes and overall agricultural productivity. Early detection and accurate classification of these diseases are essential for effective disease management strategies. By developing a robust classification system for cotton leaf diseases, farmers can promptly identify the specific diseases affecting their crops and take appropriate actions to mitigate their impact. Furthermore, predicting yield based on disease classification can assist farmers in optimizing their crop management practices, such as adjusting irrigation schedules, fertilizer application and implementing targeted pest control measures. This

can result in improved efficiency, reduced costs, and increased overall cotton yield, thereby positively impacting growing cotton may be profitable.

1.2. Problem Statement

The profitability of cotton growing has dramatically increased by the occurrence of cotton leaf diseases, which lead to substantial yield losses. However, the lack of a reliable classification system for these diseases hampers farmers' ability to identify and manage them promptly. This knowledge gap poses a significant problem for cotton farmers who strive to optimize their crop management practices and mitigate disease-related yield losses. Additionally, the absence of accurate yield prediction based on disease classification further exacerbates farmers' challenges in making informed decisions regarding resource allocation and optimizing their overall crop management strategies. The absence of a robust cotton leaf disease classification and yield prediction system limits the potential for cost-effective disease management strategies, leading to reduced profitability for cotton farmers. Farmers resort to generalized treatment approaches without a proper disease identification and prediction framework, which incur higher production costs and contribute to pesticide overuse, exacerbating environmental concerns. Thus, an urgent need is to develop an accurate and efficient system that can classify cotton leaf diseases and predict yield based on disease profiles, enabling farmers to adopt targeted management practices and optimize their resource allocation for improved crop management efficiency and economic sustainability.

1.3. Research Objective

The research objective of the study titled "Ant Colony Optimization-assisted Support Vector Machines for Accurate Cotton Leaf Disease Classification and Yield Prediction" is to develop and evaluate a novel approach that combines Ant Colony Optimization (ACO) algorithms with Support Vector Machines (SVM) to achieve accurate classification of cotton leaf diseases and reliable prediction of crop yields. Building upon the problem statement related to Motivation, where the lack of a reliable cotton leaf disease classification system hampers farmers' ability to identify and manage diseases effectively, the research objective aims to address this challenge by leveraging the power of ACO and SVM.

The specific objectives are as follows:

- Develop an ACO algorithm tailored for classifying cotton leaf diseases: Design an ACO algorithm that efficiently explores the feature space and selects the most informative and discriminative features for accurate disease classification. The ACO algorithm will be customized to prioritize selecting features relevant to cotton leaf diseases, considering their impact on disease identification and management.
- Implement a Support Vector Machines (SVM) model for cotton leaf disease classification: Employ SVM, a powerful machine learning technique, to build a classification model using the selected features obtained from the ACO algorithm. The SVM model will be trained on a labelled dataset of cotton leaf disease samples to learn the patterns and characteristics of different diseases.
- Integrate the ACO-SVM framework for accurate disease classification and yield prediction: Develop a framework that integrates the ACO algorithm with the SVM model to classify cotton leaf diseases accurately. The framework will also incorporate yield prediction based on disease classification, allowing farmers to anticipate crop productivity based on disease profiles.
- Evaluate the performance of the proposed approach: Analyze how well the ACO-SVM architecture classifies diseases and how well it predicts harvests. Compare the results with existing methods and evaluate their effectiveness in assisting farmers in making informed decisions regarding disease management strategies and resource allocation.

2. LITERATURE REVIEW

"L1-Norm Minimization Extreme Learning Machine" [26] extracts essential features from leaf images, reducing data dimensionality. ELM acts as a classifier, distinguishing healthy and diseased leaves based on these features. The process involves training the ELM model with labelled leaf images and then using it to classify unseen test images. L1-norm minimization simplifies feature extraction, while ELM offers fast and accurate classification. "Apple Plant Diseases Detection" [27]involves training the CNN model with a large dataset of labelled leaf images comprising healthy and diseased samples. The CNN learns to automatically extract relevant features from the images through convolutional and pooling layers. These known features are then fed into fully connected layers for classification. During testing, unseen leaf images are inputted into

the trained CNN, which predicts the presence of diseases based on the learned patterns.

"Restructured Deep Residual Dense Network" [28] involves training the DRDN model using a large dataset of labelled tomato leaf images, encompassing healthy and diseased samples. The DRDN consists of dense blocks and skip connections, allowing for effective feature extraction and information flow. The model learns to capture intricate patterns and disease-specific characteristics from the input images during training. In the testing phase, unseen tomato leaf images are inputted into the trained DRDN, which accurately classifies them into different disease categories. The restructured DRDN demonstrates high performance in disease identification, showcasing its potential as a reliable and efficient solution for automated tomato disease detection. The dense connections within the network enable effective feature reuse while the aid of the residual link in mitigating the vanishing gradient problem. "Precision Agriculture" [29] involves training CNN models using a dataset of leaf images, which includes both healthy and diseased samples. The CNNs are designed to learn and extract relevant texture features from the input images through layers of pooling, convolutional processing, and non-linear activation functions. The extracted features are then fed into fully connected layers for classification. By leveraging the power of CNNs, the proposed approach demonstrates high accuracy in classifying and distinguishing various leaf diseases.

"Robust Multi-Model Ensemble Method" [30] involves training multiple deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), with a diverse dataset of plant images encompassing both healthy and diseased samples. Each model specializes in capturing different aspects of disease-related features from the images. The ensemble technique combines the predictions of these individual models to obtain a final decision. This multi-model ensemble strategy improves the overall robustness and accuracy of disease detection. By leveraging the strengths of different deep learning architectures, PlantDet demonstrates a reliable and efficient solution for automated plant disease detection. The method's ability to handle diverse datasets and exploit complementary information from multiple models enhances its effectiveness in real-world scenarios. "Ensembled Transfer Learning and Multiple Kernel Learning" [31] combines ensembled transfer learning and numerous kernel learning techniques. Transfer learning is employed to leverage pre-trained models on a large-scale general dataset and fine-tune them on the specific task of CAD detection using PCG signals. Multiple kernel learning is then utilized to fuse the outputs of various classifiers trained on different feature representations of the PCG signals. This ensemble of classifiers captures diverse aspects of the data, improving the robustness and accuracy of CAD detection. The combination of transfer learning and multiple kernel learning enables the model to effectively leverage the knowledge learned from a public dataset while adapting to the specific characteristics of PCG signals related to CAD.

"Classification of Date Palm White Scale Disease" [32] involves using various machine learning algorithms to classify the disease at different stages. The framework incorporates feature extraction techniques to extract relevant information from the input data. Support Vector Machines (SVMs), Random Forests and Decision Trees all use these extracted characteristics as input into their respective models. Each algorithm specifies the disease at a specific stage, enabling a stage-wise classification approach. The framework's ability to leverage multiple machine-learning algorithms enhances the accuracy and robustness of the disease classification process."Deep Metric Learning" [33] uses deep neural networks to build feature representations with discriminative power from small data samples. To this end, the deep metric learning framework prioritizes group-level similarity while discouraging similarity across different groups. The model can effectively generalize and classify unseen citrus disease samples by training the network on a limited dataset. The approach leverages the power of deep learning to capture complex patterns and variations in citrus diseases, enabling accurate classification even with sparse data."Grape Leaf Diseases Identification System" [34] utilizes Convolutional Neural Networks (CNN) and Long Range (LoRa) technology. CNN is employed for extracting relevant features from grape leaf images.LoRa technology transmits data wirelessly to a central server for analysis and decision-making. The system's working mechanism involves capturing grape leaf images, which are preprocessed and fed into the CNN for feature extraction and classification. The trained CNN model can then classify the input images into different disease categories, and the results are transmitted through the LoRa technology to the

central server for further analysis. "GANs-Based Data Augmentation" [35] involves training a Generative Adversarial Network(GAN) to generate synthetic citrus disease images that simulate different severity levels. A realistic disease image generator and an image discriminator make up the GAN. The generated synthetic images are then combined with the original dataset to create an augmented dataset with a diverse range of disease severity levels. Deep learning models are trained on this expanded dataset to detect and classify the severity of citrus diseases. By leveraging GANs for data augmentation, the approach enhances the performance and robustness of the severity detection system, enabling accurate identification of disease severity levels in citrus crops.

"Improved Multi-Scale Feature Fusion Network" [29] involves extracting multi-scale features from apple leaf images using a deep neural network. The MSFFN incorporates multi-scale convolutional layers and feature fusion modules to capture fine-grained details and contextual information from the images. The network is trained on a large dataset of labelled apple leaf images, encompassing different disease categories and sub-classes. By leveraging the improved MSFFN, the system achieves high accuracy in recognizing and categorizing apple leaf diseases into specific sub-classes. The network's ability to capture and fuse multi-scale features effectively enhances its performance and robustness in disease recognition. This approach demonstrates its potential for practical implementation in precision agriculture, aiding in the early detection and management of apple leaf diseases. "PCA DeepNet" [36] involves combining Principal Component Analysis (PCA) with a deep neural network. Dimensionality reduction is achieved by PCA, extracting the most informative features from the tomato leaf images. The reduced-dimensional data is then fed into a deep neural network, which learns to classify the images into different disease categories. The PCA DeepNet model is trained on a dataset of labelled tomato leaf images, encompassing various disease types. By leveraging the synergistic power of PCA and deep learning, this approach offers a practical and accurate solution for tomato leaf disease detection. The combination of dimensionality reduction and deep neural networks enables the model to capture global and local patterns in the images, enhancing its performance in disease identification.

"Random Forest (RF)" [37] is a widely used machine-learning algorithm with great potential for identifying cotton leaf disease. The method uses ensemble learning to forecast accurately by integrating many decision trees. Each tree is constructed using a different selection of input characteristics and data from the training set. During training, each tree decides how to categorize each sample, and the result is calculated by averaging the findings from all trees. This ensemble approach helps reduce overfitting and increases the algorithm's robustness. In the context of cotton leaf disease identification, RF has demonstrated high accuracy by effectively capturing the complex relationships between disease symptoms and their corresponding classes. Its aptitude for dealing with complex, non-linear data and connections makes it a powerful tool for automating disease identification and supporting sustainable agricultural practices.

"Support Vector Machines (SVM)" [38] have emerged as a powerful machine-learning technique for cotton leaf disease identification. SVM works by creating a hyperplane that optimally separates different classes of cotton leaf diseases based on the provided training data. It aims to find the decision boundary that maximizes the margin between the classes, allowing for better generalization and robustness. SVM does this by changing the raw data into a space with more dimensions and features, in which a linear decision limit is easier to see. Additionally, SVM utilizes a kernel function to handle non-linear relationships, enabling accurate classification of complex disease patterns. In the context of cotton leaf disease identification, SVM has demonstrated high accuracy in classifying different disease types based on their symptom characteristics. Its ability to handle high-dimensional data, handle nonlinearity, and provide effective decision boundaries makes it a valuable tool for automated disease identification and management in agriculture.

3. COLLABORATIVE ANT COLONY OPTIMIZATION - ASSISTED SUPPORT VECTOR MACHINES

3.1. Support Vector Machine

To model intricate connections between variables, SVM is widely recognized as a statistical ML technique. SVM excellently combines the power to generalize and the capacity to deal with the curse of dimensionality. DM and ML algorithms commonly suffer from reduced effectiveness caused by the curse of dimensionality. However, SVM has proven to be a gifted method

that can achieve exceptional results even with limited training data for the algorithm. For SVMs to perform non-linear separation, kernel functions translate problems into higher dimensions. The vast majority of frequently used models can fit into the conceptual framework provided by kernel mapping. By upscaling the training dataset's original dimension space, it is possible to translate nonlinearly separable instances (in the "input space") to separable ones (in the "feature space") that can be easily discriminated. While SVM was initially designed for classification, it has also demonstrated its worth and effectiveness in regression tasks. It is important to note that a model's generalizability improves as the margins between classes increase. To achieve generalization in SVM, one often generates a sparse vector collection that includes samples at the borders of the vectors.

Algorithm 1. Pseudocode of SVM

Step 1: Set the weight vector, w, and bias, b, to zero.
Step 2: Select a suitable kernel function, K.
Step 3: Define the regularization parameter, C.
Step 4: For each training example (xi, yi):
Compute the predicted class label, y_hat, using the decision function.
Compute the hinge loss, L.
Step 5: Update the weight vector, w, and bias, b, based on the gradient of the loss function.
Step 6: Until convergence is reached, repeat steps 4 and 5.
Step 7: Once convergence is reached, the hyperplane parameters, w and b, represent the decision boundary of the SVM.

3.2. Enriched Support Vector Machine (ESVM)

The main goal of a classification problem is to establish a relationship between a set of insert variables and the class variable(e)in a given training dataset G, denoted as $F = \{P, Q\}$. M is a t-by-c matrix representing the t input features (independent factors or predictors) and $c$samples (training instances). It's important to note that $Q\varrho B$ is valid for classification problems while $Q \subseteq B$applies to regression difficulties. For instance, let's consider a categorization problem with the training dataset $F = \{p_{sw}, p_{s+1w+1,} \dots, p_{tc,} u_1, u_2, \dots u_f\}$. This dataset consists of t input data, $c$samples, and f class values where f is greater than 1. In this context, $s = 1,2, \dots, t$ represents features, $w = 1,2, \dots, c$ represents samples, and u denotes the class value(e). To train classification models (specifically, an enhanced support vector machine model) using the training dataset, the input variables $P\varrho B^e$ are mapped into high-dimensional feature spaces α, denoted as $\alpha\varrho B^e$. Then, an "Optimal Separating Hyperplane (OSH)" is generated by minimizing the margin (hyperplane-to-nearest-data-point distances for each class) using Eq.(1).

$$sgn(\sum_{s=1}^{t} q_s \propto_s . A(p_s, p_w) + v \quad (1)$$

The "Support Vectors (SV)" are defined as $p_w = 1,2, \dots, I$. The equation known as the "LaGrange dual equation," as expressed in Eq.(2), refers to "convex quadratic programming (QP)" and is employed to determine both the coefficient $\propto_s$ and the bias v.

$$MAX(\sum_{s=1}^{t} \propto_s - \frac{1}{2}\sum_{s=1}^{t}\sum_{w=1}^{t} \propto_s \propto_w . q_s q_w . A(p_s . p_w)) \quad (2)$$

Wherein:

$$\sum_{s=1}^{t} \propto_s p_s = 0 \quad (3)$$

In this context, it is considered a support vector only if p_w satisfies the condition $0 \leq \propto_s \leq U$. The coefficient of regularisation U establishes the compromise between false positives and profits. U adjusts the trade-off between simplifying the model and minimizing training mistakes. As was previously indicated, if U is too big, the SVM may generate a model that is overfitted and places excessive weight on non-separable points. On the other hand, if U is too small, the resulting model may lack accuracy. However, the kernel function Atransforms the data into hyperplanes.

SVM can utilize various kernel functions, with radial basis functions (RBFs) and polynomials being the most popular choices. By referring to Eq.(4), this research work determines the polynomial function of degree y.

$$A(p_s . p_w) = (p_s^F . p_w + 1)^y \quad (4)$$

Additionally, the factors are disregarded, and the polynomial function becomes linear when $y = 1$. Hence, the non-linear kernel function is derived using Eq.(5):

$$A(p_s . p_w) = p_s^F p_w \quad (5)$$

However, the RBF (or Gaussian kernel) is determined by Eq.(6).

$$A(p_s . p_w) = exp(-\mu p_s - p_w^y) \quad (6)$$

The Gaussian width in the kernel function can be adjusted by the value μ, which is analogous to y in the Polynomial kernel. It determines the flexibility of the final classifier. Assuming $\mu = \frac{1}{2\epsilon^2}$, where $\in$ represents a parameter. Another type of kernel function is the sigmoid kernel, which is associated with Artificial Neural Networks (ANNs). This kernel function, introduced in 1995, can be computed using the following Eq.(7):

$$A(p_s . p_w) = \tan l(\mu p_s^F p_w + b) \quad (7)$$

where kernel factors are indicated as μ and b.

Algorithm 2: Pseudocode of ESVM

Input: Training dataset F
Output: Coefficients α and bias v
Procedure:
Step 1: Initialize matrix M with input features from F
Step 2: Map input variables P into high-dimensional feature spaces α
Step 3: Generate an Optimal Separating Hyperplane (OSH) by minimizing the margin using Eq.(1)
Step 4: Define Support Vectors (SV) as p_w=1,2,..., I
Step 5: Use the LaGrange dual equation (Eq. 2) to determine α and v
Step 6: Return α and v

3.3. Setting parameters in ESVM

Finding the optimal values for critical parameters is crucial when developing a classification model. This ensures practical training and accurate assessment of the resulting classification model(e). In the case of SVM models, their performance and classification abilities depend on empirically determined parameters. Therefore, this study employs a scientific parameter exploration technique to identify the best possible parameter values. Parameters such as U (limit on the Lagrangian multipliers), μ (conditioning parameter in DM), A (the kernel), and ϱ play a significant role in the SVM model construction. The tolerance margin, represented by ϱ, defines the range within which no penalty is incurred for classification errors. Selecting the optimal values for these parameters often involves a time-consuming and exhaustive trial-and-error process. However, this research uses a Cross-Validation approach to fine-tune all the necessary parameters for building a reliable SVM model.

Algorithm 3: Parameter-Searching Strategy

Step 1: Choose the starting kernel type from options such as Polynomial, Gaussian, Linear, RBF, or Sigmoid.
Step 2: Utilize the Cross-Validation Technique to find the optimal values for parameters $U, \ni$ and ϱ. Record the corresponding accuracy measure. The grid search technique combines the total accuracy log_2 measured in logarithmic space.
Step 3: Parameters are selected using 2-fold cross-validation.
Step 4: Ten-fold cross-validation is performed using random variables.
Step 5: If improved parameter values are found, they become the new baseline for another round of 10-fold cross-validation.
Step 6: Iterate through Steps 4 and 5 until no further improvements can be made to the parameters or until the parameters reach the limits of the grid.
Step 7: If there are other available kernels, proceed to Step 8. Otherwise, go to step 4.
Step 8: Determine the optimal kernel and parameters based on the best-achieved performance metric.
Step 9: Apply the selected kernel and the parameter settings obtained in Step 4 to train the final model.
Step 10: Use the ESVM model built in Step 5 to classify new data samples.

3.4. Assignment of classes in EMSVM

Multinomial classification, or multi-class classification, involves assigning each instance in a dataset to one of three or more classes in YC and CZ. In contrast, binary classification entails categorizing instances into one of two classes. It is essential to differentiate between multi-class and multi-label classifications, where multiple labels

can be predicted for each instance. File system operations can have overlapping categories in certain cases, as seen in the YG category. Multiple programs may utilize the same file system portions, leading to several associations with system files. While many classification strategies are designed for binary classes, some are inherently binary, such as support vector machine analysis. However, there are several ways to convert SVM into a multi-class classifier.

The most common approach is the "One versus the Rest" method. In this strategy, a classifier is trained independently for each class using its training samples, which are treated as positive. To utilize this method, the underlying classifiers must compute a real-valued confidence score and assign a simple class label. However, a standard one-vs-rest approach with SVM can reduce accuracy due to the rejection region problem. This study proposes a kernel metric-based approach to address this issue.

If only one valid decision function (YE)exists, all samples will fall into region A, allowing for straightforward classification. If none or more than one of the YE functions is valid, the samples will be located in the rejection regions V or $U$for each YE function. Even if multiple YE functions are valid, all instances will be outside the rejection region V. However, suppose all YE functions are deemed unacceptable. In that case, the instances will be in the rejection region U. The conventional one-vs-rest method may fail when instances fall into a rejection zone. To handle this situation, this research utilizes the computed space proximity between samples and the associated YE functions to classify these samples.

If a case x^*falls within the rejection region V, the distance between the point of interest x^* and the best admissible hyperplane f is denoted as $e(x^*)$. This separation can be determined using Eq.(8).

$$e_f(x^*) = \frac{|Y_f(x^*)|}{\Omega_f} \quad (8)$$

In the given context, Ω_f represents the norms of the average vector of the best hyperplane for class f. Finally, this research determines the YE (decision function) of the f -best hyperplane using Eq.(9).

$$YE_f(x^*) = \sum_{i=1}^{t_{er}^f} \ni_{fe} \; q'_{fe} A(x_{fe}.x^*) + v_f \quad (9)$$

In the given scenario, t_{er}^f represents the number of errors ERe in the f-th best hyperplane, while $\ni_{fe}$ represents the Lagrange multiplier of the r -th ERe in error in the f-th best hyperplane. The $1 \le e \le t_{er}^f$ the value represents these quantities' summation. The actual class of the r-th error ERin the f-th best hyperplane is denoted as q'_{fe}, and its eigenvector is represented as x_{fe}. The kernel function's value, denoted by $A(x_{fe}.x^*)$, ranges between x^* and x_{fe}. Additionally, the divergence of the f-th hyperplane is v_f. An example's probability of incorrect classification increases closer to the decision surface. To determine which group an example belongs to, Eq.(10) is used.

$$x^* \varrho \arg(max_f(E_f)) \quad (10)$$

In contrast, if the example is outside the acceptance region U, the distance to all hyperplanes (e)must be calculated. When an example is near the decision surface, it has a higher chance of belonging to the associated class on the other side of the surface. This research work uses Eq.(11) to place the considered factor into the class closest to the rejection region.

$$x^* \varrho \arg(min_f(E_f)) \quad (11)$$

Algorithm 4: Assignment of classes in ESVM

Step 1:	Multinomial classification involves assigning instances to three or more classes.
Step 2:	Differentiate between multi-class and multi-label classifications
Step 3:	Apply the "One versus the Rest" method by training a classifier independently for each class
Step 4:	Compute a real-valued confidence score and assign a class label
Step 5:	Handle instances falling into rejection regions using computed space proximity between samples and associated decision functions
Step 6:	Calculate the distance and determine the class for instances falling within the rejection region V or outside the acceptance region U

3.5. COLLABORATIVE ANT COLONY OPTIMIZATION

3.5.1. Informative heuristics

To lessen the number of unnecessary features, the CACO-assisted feature selection (FS) approach takes advantage of the correlation among features. Eq.(12) provides the definition, wherein m is the total number of instances and G_{sa} is the quantity of characteristic $s$insisted for a. In CACO-SVM, Eq.(13) defines the heuristic data.

$$sim(G_s, G_w) = \left| \frac{\sum_{a=1}^{m} G_{sa}, G_{wa}}{\sqrt{\sum_{a=1}^{m} G_{sa}^2} \sqrt{\sum_{a=1}^{m} G_{wa}^2}} \right| \quad (12)$$

$$\alpha(G_s) = 1 - \frac{1}{|G^e|} \sum_{G_w \omega G^e} sim_{soft}(G_s, G_w) \quad (13)$$

wherein G_s is the characteristic s to be picked, $G^e$is the feature set that was selected, and $|G^e|$ is the total number of features that were ultimately chosen. sim_{soft} applies softmax normalization on the reliability coefficient sim to refine the gap between the two values by comparing the candidate feature to the selected characteristics and increasing the heuristic value in Eq.(2) when the relationship between the two is low.

3.5.2. Pheromone Initiation

The pheromone-accumulation process is the ant colony's method of education. Because the pheromone is often initialized to a fixed value, the likelihood of access to every given node is the same. In a high-dimension dataset, however, accessing many unrelated characteristics would lengthen the algorithm's execution time. Therefore, the ant colony's first exploration direction must be determined using the filter approach. In CACO, relationships are evaluated between a feature (in this case, G) and the label (in this case, U) using the symmetrical uncertain (SU), which would be frequently employed with GE and Eq.(14) defines it.

$$SU(G, U) = \left[\frac{L(G) - L(G|U)}{L(G) + L(U)} \right] \quad (14)$$

The volatility of G is $L(G)$, and the entropy of G is $L(G|U)$ if and only if U is true. More significant the feature's SU, the stronger the relationship. Additionally, we employ softmax to standardize the $SU(SU_{soft})$ that serves as the baseline for pheromone β_1.

3.5.3. Path Construction

The search space is so vast the rules for building paths have been written in a greedy and probabilistic form to account for state transitions. This strategy can strike a good mix between a comprehensive path search and a focused one on the most available direct route. By using Eq.(15), the ant uses the greedy approach to select the next feature G_a:

$$G_a = \begin{cases} argmax\{[\beta(G_o)]^{\delta}[\alpha(G_o)]^{\gamma}\} & x \le x_k \\ G_o \omega w^d \end{cases} \quad (15)$$

Characteristics G_o have an associated pheromone intensity value of $\beta(G_o)$, where W^d is a collection of all unselected features. The relative weight of pheromone data against heuristic data is set by the values of δ, γ. A greedy probability level is determined by the product of the random number$x$inside the interval [0,1] and the constant x_k. As shown in Eq.(16), the probabilistic approach determines the likelihood of selecting each characteristic from the entire pool of candidates. Then, a single characteristic is chosen using a roulette system.

$$M(G_s) = \begin{cases} \frac{[\beta(G_o)]^{\delta}[\alpha(G_o)]^{\gamma}}{\sum_{G_o \omega W^d}[\beta(G_o)]^{\delta}[\alpha(G_o)]^{\gamma}}, if \;\; G_s \omega \, W^d & if \; x > x_a \\ 0, \quad otherwise \end{cases} \quad (16)$$

3.5.4. New-fangled Pheromone

Pheromone updates often employ two classic ant systems, exceptional AS (EAS) and Max-Min AS (MxMnAS). In the first phase of the iteration, the feature subset built by each ant on the high-dimensional data dataset varies greatly. In addition, if all the ants participate in the pheromone updating, the pheromone deposition will be irrational. Consequently, the MxMnAS update procedure is used in this research. Overaccumulation of pheromones might lead to a local optimum. Thus, only a tiny quantity is stored at each repetition. After f cycles, the pheromone of features G_s is updated as described by Eq.(17).

$$\beta_{f+1}(G_s) = (1 - \varphi)\beta_f(G_s) + h\Delta\beta_f^v(G_s) \quad (17)$$

$$\Delta\beta_f^v(G_s) = \begin{cases} figness(F), & if \; G_s \omega J \\ 0, & otherwise \end{cases} \quad (18)$$

wherein φ is the evaporation factor for pheromones, which prevents the unlimited buildup of pheromones and allows the algorithm to reject the previously generated common pathways swiftly. The pheromone's iterative growth in size is controlled by a weight constant of f. As the buildup of pheromone $\Delta\beta_f^v(G_s)$, where J denotes the subset of features generated by the modern optimum ant, CACO utilize the fitness value of the ant with the most incredible fitness of iteration f, fitness(J).

3.5.5. Fitness Function

The fitness function incorporates the precision of K-closest neighbour (KNN) and distances with a weight π defined as in Eq.(19).

$$fitness = (accuracy + \pi.distance) \quad (19)$$

The cross-validation technique is applied to the training data to get the predicted value. To deal with high-dimensional datasets with imbalanced categories. It claims the balancing correctness is given in Eq.(20) as the initial item.

$$balanced\_accuracy = \frac{1}{u}\sum_{s=1}^{u} FMB_s \quad (20)$$

wherein u is the total number of data categories and FMB_s is the percentage of correctly labelled samples score s.

CACO builds the distance metric as a balance indication on Eq.(21) in contrast to the designed technique. It is desirable to maximize the distance between S_s, and the closest sample of a different class and minimize the mean of the lengths between all instances of the same class in a collection of instances of class d.

$$balanced\_distance = \frac{1}{u}\sum_{d=1}^{u}\frac{1}{|S^d|}\sum_{S_s \omega S^d} distance_{S_s} \quad (21)$$

$$distance_{S_s} = -\frac{1}{|S^d|-1}\sum_{S_w \neq S_s, S_w \omega S^d} Dis(S_s, S_w) + minDis(S_s, S_a), \quad S_s \omega S^d,\ S_a \notin S^d \quad (22)$$

where the count of objects in class d is denoted by the symbol $|S^d|$. $Dis(S_s, S_w)$ The function uses the Manhattan distance to determine the separation of two instances. All data should be resized to the range [0,1] to remove the impact of the chosen number of attributes and the variety of attribute values. Subtract the chosen feature count from the Manhattan distance after calculating it.

Algorithm 5: Pheromone-based Fitness Evaluation

Step 1: Informative Heuristics

- Let m be the total number of instances.
- For each characteristic s
- Calculate the quantity G_{sa} of characteristic s
- insisted for a.
- Calculate the correlation sim(G_s, G_w) between feature G_s and feature G_w using Eq.(1).
- Calculate the heuristic value $\alpha(G_s)$ using Eq.(2).

Step 2: Pheromone Initiation

- For each feature G:
- The relationship between feature G and label U is evaluated using symmetrical uncertain (SU) defined in Eq.(3).
- Standardize SU using softmax to obtain SU_{soft}.
- Initialize the baseline pheromone β_1 for feature G.

Step 3: Path Construction

- While constructing the feature subset:
- Select the next feature, G_a using a greedy approach based on pheromone and heuristic values defined in Eq.(4).
- If a probabilistic approach is required (based on the random number x and constant x_k), select a single character using a roulette system with probabilities $M(G_s)$ defined in Eq.(5).

Step 4: New-fangled Pheromone

- After each iteration f
- Update the pheromone β_f for each feature G_s using Eq.(6).
- Calculate the fitness value fitness for each ant using Eq.(8).
- Determine the subset J of features generated by the ant with the highest fitness.
- Update the pheromone increment $\Delta\beta_f^{v(G_s)}$ for each feature G_s using Eq.(7).

Step 6: Function of Fitness

- Calculate the fitness value for each ant using the precision of with weight π defined in Eq.(8).
- Apply cross-validation technique to the training data to obtain the predicted value.
- Calculate balanced_accuracy using Eq.(9).

- Calculate balanced_distance using Eqs.(10) and Eq.(11).

Output:
- Return the subset of features that have historically achieved the best fitness.

3.5.6. Feature Count Identification

Before it can figure out how many access nodes there are, the ant has to know how many features it has selected. The most reliable approach is using the CACO algorithm to discover the same feature under each parameter and then comparing these subsets of features to arrive at the final subset of features. The CACO method must be run T times, wherein T is the overall features, which is quite time-consuming. The estimation of classification accuracy is done by adding a feature at different time intervals. They cease constructing when the model is no longer promoted for consecutive times. In those methods, the CACO procedure must be executed at least once, assuming the feature number is a. Unfortunately, for each cycle, each ant must compute the fitness function about $(t + a)$ times. The above solutions do not adequately take into account or make use of GE particular qualities. Assuming that the ant colony could construct the optimal subset within the constraints of the required feature count, CACO divided the GE into three intervals and named them correlation, redundant, and uncorrelated.

Classification performance can be improved by integrating non-redundant related features in the correlation interval. During this period, the feature subset generated by ACO contains multiple duplicate features to maintain high classification accuracy. The presence of duplicate features does not affect the performance, resulting in smooth and consistent classification accuracy. As the ACO algorithm includes all possible redundant features, the feature subset transitions into the uncorrelated region. Due to the many characteristics considered for inclusion in the algorithm, irrelevant features are incorporated here. Consequently, the classification performance starts to decline. The size of the interval (IntSize) is determined by Eq.(23). By solving Eq.(24), CACO can determine that the number of characteristics of the right endpoints of the $i$intervals is $(PoiNum_i)$. Multiple equivalent ideal values might exist, but CACO does not compute all possible terminals and select the best one. Varying the number of endpoint characteristics from few to many forces a recalculation of the best value. When the ideal value stops increasing after some number of repetitions (a), the procedure is terminated, and the optimal amount of features is returned $(PoiNum_{optimal})$.

The new interval's endpoints are the points $(PoiNum_{optimal-1}\ and\ PoiNum_{optimal+}\ )$ immediately to the left and right of the ideal endpoint. Furthermore, the number of features at the new i interval's right endpoint $(PoiNum_i)$is calculated with Eq.(25). Even if CACO are just interested in finding a limited number of discrete points, this gradually decreasing interval size guarantees a thorough global and local search. And the new range will be subdivided even more. Split the abovementioned intervals until the maximum allowed iterations (MaxIte) are attained. The term$a$ will grow with each interval refresh since the more extensive the gap between updates, the more noticeable the gap between updates becomes.

$$IntSize = \frac{FeaNum}{IntNum} \tag{23}$$

$$PoiNum_i = IntSize * i, \quad i = 1,2,3 \ldots\ldots \tag{24}$$

$$PoiNum_i = \frac{i}{IntNum} * \left(PoiNum_{optimal+1} - PoiNum_{optimal-1}\right) + PoiNum_{optimal-1}, \quad i = 1,2,3 \ldots. \tag{25}$$

Algorithm 6: Feature Count Identification (CACO-FCI)

Input:
- FeaNum: Total number of features
- IntNum: Number of intervals
- MaxIte: Maximum number of iterations

Output:
- $PoiNum_{optimal}$: Optimal number of features

Procedure:

Step 1: Initialize IntSize = FeaNum / IntNum
Step 2: Initialize $PoiNum_{optimal}$ = FeaNum (default value)
Step 3: Set a = 0
Step 4: Repeat the following steps until the maximum allowed iterations (MaxIte) is reached:
- **a)** Increment a by 1
- **b)** Calculate $PoiNum_{left}$ = $PoiNum_{optimal}$ - 1

c) Calculate $PoiNum_{right} = PoiNum_{optimal} + 1$
d) Set $PoiNum_i = PoiNum_{left}$
e) For i = 1 to IntNum:
f) Calculate $PoiNum_i = (i / IntNum) * (PoiNum_{right} - PoiNum_{left}) + PoiNum_{left}$
g) Find the maximum classification accuracy for each PoiNum_i
h) If the maximum classification accuracy stops increasing after a certain number of repetitions (a), terminate the procedure
i) Update PoiNum_optimal to the value with the highest classification accuracy
j) Update IntSize = FeaNum / IntNum
k) Divide the intervals further by recalculating PoiNum_i using the new $PoiNum_{optimal}$

Step 5: Return $PoiNum_{optimal}$ as the optimal number of features.

3.5.7. Reduction of Feature

Iteratively decreasing the interval size until it equals one or evaluating each feature value in the final interval will yield the optimal sample size. This, however, is unnecessary and will lengthen the whole production. There is a sharp break as the number of characteristics selected approaches infinite in FS. The capacity of CACO to create OptiFS becomes the determining factor in the classification accuracy rather than the feature count. First, we use the preceding methodology to estimate the confidence interval. The maximum interval value is used as a proxy for the ideal feature number, which is found with fewer interval splits. Then, in the second phase, CACO employs the search ability to identify the total amount of features included in the search evolution.

CACO employ a multi-generational approach to feature reduction. OptiFS chose J for the modern population because of its high fitness. Then, using Eq.(26), CACO determines the likelihood that each characteristic in $J$and it will be chosen as the reducing feature.

$$M(G_s) = \begin{cases} \frac{1 - SUsoft(G_s)}{\sum_{G_o \omega J}[1 - SUsoft(G_o)]} & if\ G_s \omega J, \\ 0, & otherwise \end{cases} \quad (26)$$

In general, a lower correlation means that a feature is more likely to be chosen as the characteristic to be eliminated. Using the roulette approach, CACO select a feature G_w to remove from the characteristic subset J before determining the fitness of the new feature subset G_{new}. Feature subset J may be replaced with G_{new}if the fitness of G_{new} is greater than the ideal historical fitness $fitness_l$. The preceding steps will be performed m_{max} times until J can no longer be updated. CACO constrain m_{max} to be no more than (bm)bits in size relative to the J-bit feature subset. The final stage is incorporating the smaller group of features into the pheromone update. The offspring ants produce a feature subset of the same size as the reduced subset. The complete feature reduction procedure is displayed in Algorithm 7.

Algorithm 7: Feature Reduction using CACO (CACO-FR)

Input:
- Dataset: The input dataset with m instance features
- T: Number of iterations for the CACO algorit
- m_max: Maximum number of iterations for reduction
- b: Factor controlling the size of the decreased subset

Output:
- Reduced feature subset

Procedure:

Step 1: Initialization
- Initialize the pheromone matrix β with a fixe for all features.

Step 2: Feature Subset Selection using CACO
- Initialize an ant colony with T ants.
- Construct feature subsets using the ant col following the CACO algorithm.
- Evaluate the fitness of each feature subset using a fitness function based on classification accuracy.
- Update the pheromone matrix β based on the of the feature subsets.

Step 3: Feature Reduction using OptiFS
- Estimate the confidence interval for the ideal number:
  a) Determine the maximum interval
- Choose the best feature subset:
  a) Select the feature subset with the highes (OptiFS) from the last interval.
  b) Iterate feature reduction using m_max ite
  c) Determine the likelihood of each feature in OptiFS to be chosen as the reducing feature using Eq.(15).
  d) Select a feature G_w to remove from OptiF

the roulette approach.

e) Determine the fitness of the new feature G_{new}.

f) If the fitness of G_{new} is greater than the ideal historical fitness, replace OptiFS with G_{new}.

g) Limit the size of m_max to be no more than (b * m) bits in size relative to the original feature subset.

- Include the decreased feature subset pheromone update:
  a) Update the pheromone matrix β with the decreased feature subset generated by the progeny ants.

Step 4: Output

- Return the reduced feature subset obtain the feature reduction process.

4. ABOUT THE DATASET

The "Cotton Plant Disease Dataset" is a comprehensive collection of images focusing on diseases that affect cotton plants. The dataset comprises 26,100 high-resolution images, making it a valuable resource for researchers, plant pathologists, and data scientists in cotton plant health. With approximately 4GB, this dataset provides an extensive range of images capturing various stages and manifestations of cotton plant diseases. The dataset includes four primary diseases: Aphids, Armyworms, Bacterial Blight, and Powdery Mildew. Additionally, a subset of the dataset contains images of healthy cotton leaves to facilitate comparison and analysis. It is important to note that the dataset primarily focuses on diseases affecting the leaves of cotton plants. Therefore, it does not include reference images for diseases occurring on cotton plants' stems, buds, flowers, or bolls. This dataset offers an excellent opportunity for researchers and practitioners to develop and evaluate machine learning algorithms, computer vision models, and other analytical techniques to automate identifying and classifying cotton plant diseases. By leveraging this dataset, researchers can enhance disease detection and monitoring methods, leading to more efficient disease management strategies and improved crop yield in cotton farming. The dataset is available at https://www.kaggle.com/datasets/dhamur/cotton-plant-disease.

5. PERFORMANCE METRICS

- **Classification Accuracy (CA)** is a performance metric used to assess the precision of a classification model in correctly identifying different types of leaf diseases affecting cotton plant leaves. It is calculated as the ratio of correctly classified diseased leaves to the dataset's total number of diseased leaves.
- **F-Measure (FM)** Disease detection in cotton leaves is measured using a composite metric that considers both accuracy and reliability. It provides a single number representing the model's ability to accurately detect and categorize illnesses impacting cotton plant leaves by quantifying the harmonic mean of accuracy and recall.
- **Fowlkes-Mallows Index (FMI)** is a statistical measure used to evaluate the similarity between two different methods or algorithms in their ability to correctly identify and classify different types of diseases in cotton plant leaves. It quantifies the agreement between the pairwise similarities of the disease identification results, providing a single value representing the similarity or agreement level between the two methods.
- **Matthews Correlation Coefficient (MCC)** is a statistical measure used to evaluate a classification model's efficacy by weighing the number of correct and incorrect classifications. It provides a single value representing the correlation between the predicted and actual disease classifications, considering both the model's sensitivity (recall) and specificity.

6. RESULTS AND DISCUSSION

6.1. Assessment of Classifiers using CA and FM Performance Metrics

Figure 1 presents a comparative analysis of three classification algorithms: RF, SVM, and CACO-SVM. The analysis is based on two evaluation metrics: classification accuracy (CA) and F-measure (FM).

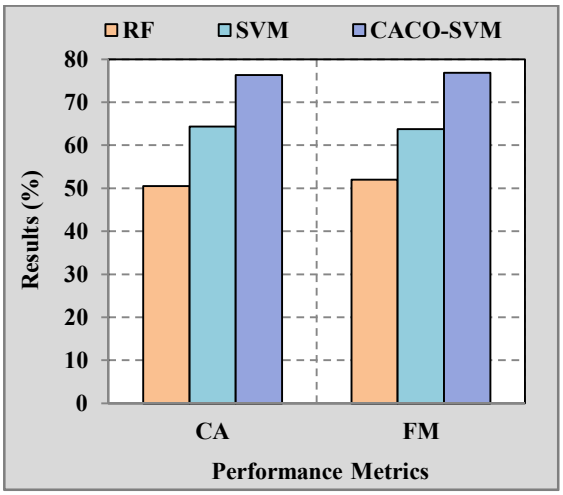

Figure 1. CA and FM

Classification accuracy (CA) measures the ability of a classification model to classify instances in the dataset correctly. It is expressed as a percentage, indicating the proportion of accurately classified instances. In the given analysis, RF is an ensemble learning method that combines multiple decision trees to make predictions. Each tree is built on a random subset of features and uses a voting mechanism to determine the final classification. RF achieves a CA of 50.487% in the analysis. The lower CA score could be attributed to several factors, such as the limitations of individual decision trees or the suboptimal combination of features in this specific dataset. SVM is a robust supervised learning algorithm used for classification and regression tasks. SVM aims to find an optimal hyperplane that separates different classes while maximizing the margin. SVM achieves a CA of 64.333% in the analysis. The higher CA score indicates that SVM successfully found a decision boundary that performs better than RF in classifying the instances in the dataset. CACO-SVM combines the principles of SVM with an enhanced version of ACO, namely Collaborative ACO (CACO), to enhance its performance. CACO is a metaheuristic algorithm inspired by the behavior of ant colonies. It helps to improve the feature selection process and find more optimal support vectors for SVM. CACO-SVM achieves the highest CA of 76.344% in the analysis. The collaborative optimization assisted by ant colony behavior allows CACO-SVM to identify better support vectors and improve the overall accuracy of the classification.

The F-measure combines precision and recall into a single metric and provides a balanced assessment of the model's performance. RF achieves an FM of 52.013% in the analysis. The lower FM score suggests that RF may have lower precision and recall in this classification task. It might struggle with accurately identifying positive and negative instances due to decision tree-based ensemble learning limitations. SVM achieves an FM of 63.757% in the analysis. The higher FM score implies that SVM exhibits better precision and recall than RF. SVM's ability to find an optimal hyperplane allows it to effectively separate different classes and reduce false positives and negatives, leading to a higher F-measure. CACO-SVM achieves the highest FM of 76.819% in the analysis. The incorporation of collaborative ant colony optimization in SVM helps to enhance feature selection and improve the identification of support vectors. As a result, CACO-SVM achieves higher precision and recall, leading to an improved F-measure compared to both RF and SVM.

The working mechanisms of the classification algorithms contribute to the observed CA and FM results in Figure 1. RF and SVM have their strengths and limitations, but CACO-SVM leverages the collaborative optimization strategy inspired by ant colonies to enhance the performance of SVM. This allows CACO-SVM to achieve higher classification accuracy and a more balanced F-measure, making it the most effective algorithm among the three for the given classification task.

Table 1.CA and FM Results

Classification Algorithms	CA	FM
RF	50.487	52.013
SVM	64.333	63.757
CACO-SVM	76.344	76.819

6.2. Assessment of Classifiers using FMI and MCC Performance Metrics

Figure 2 presents the analysis of the FMI and MCC for three classification algorithms: RF, SVM, and CACO-SVM. These metrics provide insights into the algorithms' clustering quality and overall classification performance.

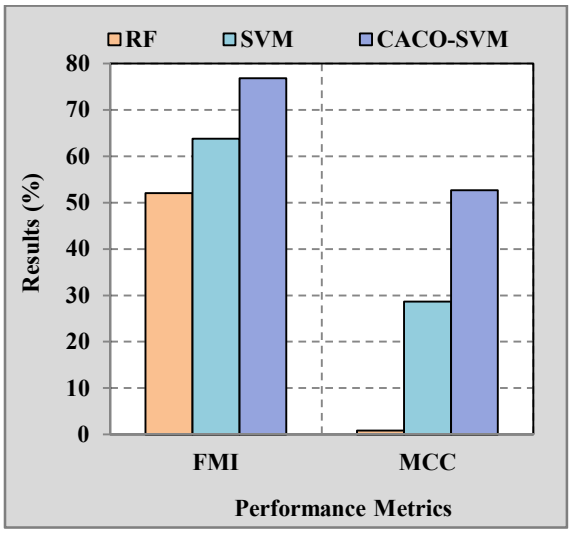

Figure 2. FMI and MCC

The FMI measures the similarity between the predicted and ground truth clusters, indicating the accuracy of the clustering results. A higher FMI score implies better clustering performance. In Figure 2, RF achieves an FMI of 52.016, SVM achieves 63.763, while CACO-SVM attains a significantly improved FMI of 76.821. The results indicate that CACO-SVM surpasses both RF and SVM in terms of clustering accuracy, suggesting that the collaborative ant colony optimization contributes to enhanced cluster formation by intelligently exploring the feature space.

The MCC considers true positives, false positives, and false negatives, offering a comprehensive measure of overall classification performance. Higher MCC values indicate better classification accuracy. In Figure 2, RF achieves an MCC of 0.887, SVM achieves 28.669, whereas CACO-SVM achieves a notable MCC of 52.677. These results demonstrate that CACO-SVM outperforms RF and SVM regarding classification accuracy, thanks to the synergistic collaboration between the ant colony optimization and support vector machine components.

CACO-SVM's working mechanism combines the power of collaborative ant colony optimization and support vector machines. The collaborative ant colony optimization intelligently explores the search space, optimizing the selection of features and hyperparameters in the support vector machine model. This cooperative approach allows CACO-SVM to identify the most relevant features effectively and fine-tune the SVM model, improving clustering accuracy and classification performance.

Figure 2 reveals the FMI and MCC analysis of three classification algorithms: Random Forest, Support Vector Machine, and CACO-SVM. The results demonstrate the superior clustering accuracy and classification performance of CACO-SVM. By intelligently integrating collaborative ant colony optimization with support vector machines, CACO-SVM achieves higher FMI and MCC scores, highlighting its potential as an intelligent and practical approach for clustering and classification tasks.

Table 2.FMI and MCC Results

Classification Algorithms	FMI	MCC
RF	52.016	0.887
SVM	63.763	28.669
CACO-SVM	76.821	52.677

7. CONCLUSION

This research focused on developing an innovative approach for accurate cotton leaf disease classification and yield prediction. By combining Ant Colony Optimization (ACO) algorithms with Support Vector Machines (SVM), we aimed to address cotton farmers' challenges in disease management and resource allocation. We successfully created an integrated ACO-SVM framework by developing a customized ACO algorithm for feature selection and implementing an SVM model for disease classification. This framework demonstrated promising results in accurately identifying cotton leaf diseases and predicting crop yields based on disease profiles. The evaluation of the proposed approach revealed its effectiveness in assisting farmers in making informed decisions regarding disease management strategies and resource allocation. The accurate classification of cotton leaf diseases enables targeted treatments, reducing production costs and minimizing the environmental impact of generalized approaches. By providing a reliable disease identification and yield prediction system, our research contributes to optimizing disease management practices, increasing profitability for cotton farmers, and improving sustainability in cotton production. Future research can explore further enhancements, such as incorporating additional data sources, refining the ACO algorithm, and expanding the scope to include diseases affecting other parts of the cotton plant.

Overall, this study marks a significant step toward advancing cotton farming practices and supports the goal of achieving sustainable and efficient crop management in the face of cotton leaf diseases.

REFERENCES

[1]. Chauhan, P., Mandoria, H.L., Negi, A., Rajput, R.S.: Plant diseases concept in smart agriculture using deep learning. In: Smart Agricultural Services Using Deep Learning, Big Data, and IoT. pp. 139–153 (2020). https://doi.org/10.4018/978-1-7998-5003-8.ch008.

[2]. Veerendra, G., Swaroop, R., Dattu, D.S., Jyothi, C.A., Singh, M.K.: Detecting plant Diseases, quantifying and classifying digital image processing techniques. Mater. Today Proc. 51, 837–841 (2021). https://doi.org/10.1016/j.matpr.2021.06.271.

[3]. Kundu, N., Rani, G., Dhaka, V.S., Gupta, K., Nayaka, S.C., Vocaturo, E., Zumpano, E.: Disease detection, severity prediction, and crop loss estimation in MaizeCrop using deep learning. Artif. Intell. Agric. 6, 276–291 (2022). https://doi.org/10.1016/j.aiia.2022.11.002.

[4]. Sinha, A., Shekhawat, R.S.: Review of image processing approaches for detecting plant diseases ISSN 1751-9659. IET Image Process. 14, 1427–1439 (2020). https://doi.org/10.1049/iet-ipr.2018.6210.

[5]. Linker, R.: Machine learning based analysis of night-time images for yield prediction in apple orchard. Biosyst. Eng. 167, 114–125 (2018). https://doi.org/10.1016/j.biosystemseng.2018.01.003.

[6]. Wu, H., Fang, L., Yu, Q., Yuan, J., Yang, C.: Plant leaf identification based on shape and convolutional features. Expert Syst. Appl. 219, 119626 (2023). https://doi.org/10.1016/j.eswa.2023.119626.

[7]. Wu, G., Fang, Y., Jiang, Q., Cui, M., Li, N., Ou, Y., Diao, Z., Zhang, B.: Early identification of strawberry leaves disease utilizing hyperspectral imaging combing with spectral features, multiple vegetation indices and textural features. Comput. Electron. Agric. 204, 107553 (2023). https://doi.org/10.1016/j.compag.2022.107553.

[8]. Venkatesh, J., Ramasamy, K.K., Aruna, M., Praveen Kumar Rao, K., Sasikala, N., Nasani, K.: EAgri: Smart Agriculture Monitoring Scheme using Machine Learning Strategies. In: Proceedings of the 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems, ICSES 2022 (2022). https://doi.org/10.1109/ICSES55317.2022.9914216.

[9]. Kurmi, Y., Gangwar, S.: A leaf image localization based algorithm for different crops disease classification. Inf. Process. Agric. 9, 456–474 (2022). https://doi.org/10.1016/j.inpa.2021.03.001.

[10]. Singh, V., Chug, A., Singh, A.P.: Classification of Beans Leaf Diseases using Fine Tuned CNN Model. Procedia Comput. Sci. 218, 348–356 (2023). https://doi.org/10.1016/j.procs.2023.01.017.

[11]. Senthilkumar, A., Ramkumar, J., Lingaraj, M., Jayaraj, D., Sureshkumar, B.: Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing. Int. J. Comput. Networks Appl. 10, 217–230 (2023). https://doi.org/10.22247/ijcna/2023/220737.

[12]. Ramkumar, J., Vadivel, R.: Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN). World J. Eng. 15, 306–311 (2018). https://doi.org/10.1108/WJE-08-2017-0260.

[13]. Ramkumar, J., Vadivel, R.: Performance Modeling of Bio-Inspired Routing Protocols in Cognitive Radio Ad Hoc Network to Reduce End-to-End Delay. Int. J. Intell. Eng. Syst. 12, 221–231 (2019). https://doi.org/10.22266/ijies2019.0228.22.

[14]. Lingaraj, M., Sugumar, T.N.N., Felix, C.S.S., Ramkumar, J.: Query aware routing protocol for mobility enabled wireless sensor network. Int. J. Comput. Networks Appl. 8, 258–267 (2021). https://doi.org/10.22247/ijcna/2021/209192.

[15]. Ramkumar, J., Vadivel, R.: Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network. Int. J. Comput. Networks Appl. 8, 455–464 (2021). https://doi.org/10.22247/ijcna/2021/209711.

[16]. Ramkumar, J., Samson Dinakaran, S., Lingaraj, M., Boopalan, S., Narasimhan, B., Dinakaran, S.S., Lingaraj, M., Boopalan, S., Narasimhan, B.: IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion. In: Murari, K., Prasad Padhy, N., and Kamalasadan, S. (eds.) Lecture Notes in Electrical Engineering. pp.

17–27. Springer Nature Singapore, Singapore (2023). https://doi.org/10.1007/978-981-19-8353-5_2.

[17]. J, R.: Meticulous Elephant Herding Optimization based Protocol for Detecting Intrusions in Cognitive Radio Ad Hoc Networks. Int. J. Emerg. Trends Eng. Res. 8, 4548–4554 (2020). https://doi.org/10.30534/ijeter/2020/82882020.

[18]. Ramkumar, J., Vadivel, R.: Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks. Wirel. Pers. Commun. 120, 887–909 (2021). https://doi.org/10.1007/s11277-021-08495-z.

[19]. Ramkumar, J.: Bee inspired secured protocol for routing in cognitive radio ad hoc networks. Indian J. Sci. Technol. 13, 2159–2169 (2020). https://doi.org/10.17485/ijst/v13i30.1152.

[20]. Ramkumar, J., Kumuthini, C., Narasimhan, B., Boopalan, S.: Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol. 2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022. 1–6 (2022). https://doi.org/10.1109/ICACTA54488.2022.9752899.

[21]. Menakadevi, P., Ramkumar, J.: Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data. 2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022. 1–5 (2022). https://doi.org/10.1109/ICACTA54488.2022.9753203.

[22]. Jaganathan, R., Ramasamy, V., Ramkumar, J., Vadivel, R.: Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay. Int. J. Intell. Eng. Syst. 12, 221–231 (2019). https://doi.org/10.22266/IJIES2019.0228.22.

[23]. Jaganathan, R., Vadivel, R.: Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks. Int. J. Comput. Digit. Syst. 10, 1063–1074 (2021). https://doi.org/10.12785/ijcds/100196.

[24]. Vadivel, R., Ramkumar, J.: QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications. Inc. Internet Things Healthc. Appl. Wearable Devices. 109–121 (2019). https://doi.org/10.4018/978-1-7998-1090-2.ch006.

[25]. Ramkumar, J., Vadivel, R.: CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks. In: Advances in Intelligent Systems and Computing. pp. 145–153. Springer Verlag (2017). https://doi.org/10.1007/978-981-10-3874-7_14.

[26]. Dwivedi, R., Dutta, T., Hu, Y.C.: A Leaf Disease Detection Mechanism Based on L1-Norm Minimization Extreme Learning Machine. IEEE Geosci. Remote Sens. Lett. 19, 1–5 (2022). https://doi.org/10.1109/LGRS.2021.3110287.

[27]. Vishnoi, V.K., Kumar, K., Kumar, B., Mohan, S., Khan, A.A.: Detection of Apple Plant Diseases Using Leaf Images Through Convolutional Neural Network. IEEE Access. 11, 6594–6609 (2023). https://doi.org/10.1109/ACCESS.2022.3232917.

[28]. Zhou, C., Zhou, S., Xing, J., Song, J.: Tomato Leaf Disease Identification by Restructured Deep Residual Dense Network. IEEE Access. 9, 28822–28831 (2021). https://doi.org/10.1109/ACCESS.2021.3058947.

[29]. Barburiceanu, S., Meza, S., Orza, B., Malutan, R., Terebes, R.: Convolutional Neural Networks for Texture Feature Extraction. Applications to Leaf Disease Classification in Precision Agriculture. IEEE Access. 9, 160085–160103 (2021). https://doi.org/10.1109/ACCESS.2021.3131002.

[30]. Shovon, M.S.H., Mozumder, S.J., Pal, O.K., Mridha, M.F., Asai, N., Shin, J.: PlantDet: A Robust Multi-Model Ensemble Method Based on Deep Learning for Plant Disease Detection. IEEE Access. 11, 34846–34859 (2023). https://doi.org/10.1109/ACCESS.2023.3264835.

[31]. Pathak, A., Mandana, K., Saha, G.: Ensembled Transfer Learning and Multiple Kernel Learning for Phonocardiogram Based Atherosclerotic Coronary Artery Disease Detection. IEEE J. Biomed. Heal. Informatics. 26, 2804–2813 (2022). https://doi.org/10.1109/JBHI.2022.3140277.

[32]. Hessane, A., Youssefi, A. El, Farhaoui, Y., Aghoutane, B., Amounas, F.: A Machine Learning Based Framework for a Stage-Wise Classification of Date Palm White Scale Disease. Big Data Min. Anal. 6, 263–272 (2023).

https://doi.org/10.26599/BDMA.2022.9020022.

[33]. Janarthan, S., Thuseethan, S., Rajasegarar, S., Lyu, Q., Zheng, Y., Yearwood, J.: Deep metric learning based citrus disease classification with sparse data. IEEE Access. 8, 162588–162600 (2020). https://doi.org/10.1109/ACCESS.2020.3021487.

[34]. Zinonos, Z., Gkelios, S., Khalifeh, A.F., Hadjimitsis, D.G., Boutalis, Y.S., Chatzichristofis, S.A.: Grape Leaf Diseases Identification System Using Convolutional Neural Networks and LoRa Technology. IEEE Access. 10, 122–133 (2022). https://doi.org/10.1109/ACCESS.2021.3138050.

[35]. Zeng, Q., Ma, X., Cheng, B., Zhou, E., Pang, W.: GANS-based data augmentation for citrus disease severity detection using deep learning. IEEE Access. 8, 172882–172891 (2020). https://doi.org/10.1109/ACCESS.2020.3025196.

[36]. Roy, K., Chaudhuri, S.S., Frnda, J., Bandopadhyay, S., Ray, I.J., Banerjee, S., Nedoma, J.: Detection of Tomato Leaf Diseases for Agro-Based Industries Using Novel PCA DeepNet. IEEE Access. 11, 14983–15001 (2023). https://doi.org/10.1109/ACCESS.2023.3244499.

[37]. Pavlov, Y.L.: Random forests. Random For. 45, 1–122 (2019). https://doi.org/10.4324/9781003109396-5.

[38]. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. 20, 273–297 (1995). https://doi.org/10.1007/bf00994018.