Disruptive technologies for smart farming: Tomato leaf disease recognition systems based on Machine Learning

2022

by

S. V. M. Ravichandra Reddy Kovvuri
Under the Supervision of
Dr. Abhishek Kaushik

School of Informatics and Creative Arts
Department of Computing Science and Mathematics

Acknowledgment

This study is not supported by any organization. This is intended for academic and learning purposes. I am thankful to Dundalk institute of technology for providing the Master's program which helps me work on this project. I would like to thank my supervisor Dr. Abhishek Kaushik, Lecturer for M.Sc. Data Analytics, Department of Computer Science and Mathematics, Dundalk Institute of Technology, Dundalk, Ireland for this research's encouragement, invaluable patience, and feedback.

I would like to thank the lecturers from the Data Analytics program, Dr. Jack McDonnel, Dr. Peadar Grant, Dr. Rajesh Jaiswal, and Dr. Siobhan Connolly for their lectures, projects, and laboratories. They helped me to get knowledge about different sectors.

Table of Contents Page No.

Abstract – **Food security is a major concern in any country. Farmers face many problems while cultivating plants and they must take precautions at every stage of cultivation. Plants get diseases for various reasons like bacteria, insects, and fungus. Some diseases can be detected by examining the symptoms on the leaves. Detecting the disease is a major concern and it needs an agricultural professional to examine the plants. This process is expensive and time taking. Machine learning and deep learning state of art help in image recognition and can be used to detect diseases on time without the need of an agricultural professional. In this project, the diseases in tomato leaves will be detected using image processing. The data from the images are extracted by using different vectorization methods and performing classification algorithms like logistic, KNN, and SVM on each vector data to find the best model for this problem.**

Keywords – Diseases, Machine learning, Image processing, Classification

1. Introduction

Agriculture is a major sector and backbone of any country. Different countries implement different techniques and cultivate different crops according to their environmental and geographical conditions. Some major crops are cultivated by a plethora of countries like tomatoes, potatoes, and Corn. Reduction in agriculture production will rise the imports and this will change many factors in the country. In countries like India, many farmers cannot afford for checking the disease. Even though farming is a major field for all countries, it is not advancing with the present trend. Advancements in agriculture are much needed, and some poor nations are still working with old methods. Lack of knowledge in cultivation and diseases is a major issue for many farmers. Most farmers implement traditional farming methods and are unable to detect diseases in the early stage. Machine learning and deep learning techniques are capable of detecting diseases at early stages and help farmers to deal with the diseases. The motivation for this research is to build a model which can predict the disease on the tomato leaves in the early stages. So, any farmer with minimum facilities and operatable knowledge on the computer can use this model irrespective of location, region, and time. So, farmers can use it as a primary opinion, or they can use this model to confirm their assumptions about the infected disease if they have confusion about the disease. This will ease the work of the farmers and it reduces the dependency on agricultural professionals.

Tomatoes are one of the most consumed vegetables in the world(Oishimaya Sen Nag 2020). Since it is a nutrient-dense plant and gives good income after production, many farmers are interested to cultivate these plants. Every crop is vulnerable to certain diseases and tomato plants are also infected by many diseases. If the farmer notices a disease, the farmer must wait for an agricultural professional to test the sample and confirm the result. This process will take so much time which can further deteriorate the current crop. In developing countries, there is no availability of agricultural professionals, and the farmers in remote places have to wait so long for the results. This waiting time increases the spread of the disease and increases the risk of production reduction. These issues can be reduced using machine

learning techniques. For prediction of the diseases in the tomato leaves, nine major diseases are selected, those are Bacterial spot, Early blight, Late blight, Leaf Mold, Septoria leaf spot, Target Spot, Tomato Mosaic virus, Tomato Yellow Leaf Curl Virus, and Two-Spotted Spider Mite. For this research, the images of these diseases are collected from an open website(Huang and Chang 2020). Healthy leaf pictures are also included along with these nine categories for reducing bias and detecting healthy leaves. If the healthy leaf is not included in the research, the final model is not capable of detecting the healthy leaves and giving a disease as output if healthy leaf data is provided. So, the final model can be capable to find these nine diseases and healthy plants.

Helping farmers and motivating them to use the new technologies is the aim of this project. This project can be extended to the other crops also. Image processing is the major field for this project and this project needs many tools and libraries for the analysis.

1.1 Research Objectives

This study is intended to explore and find answers to the below research objectives.

- Explore different vectorization methods for image processing and check which method is more suitable for tomato leaves disease detection.
- Perform different scaling and modeling techniques and find which method is more suitable for this problem.
- Perform different classification algorithms and evaluate the metrics to check which algorithm is good for disease detection.
- Evaluate the results and find which features are giving good results in disease detection.

1.2 Research Questions

The research questions for predicting the tomato disease on the leaves using image processing and machine learning are as follows:

1. Which vectorization techniques are suitable for predicting the disease using machine learning?
2. Is dimension reduction techniques help to detect the major part of the image and increase the accuracy?
3. Compare evaluation metrics of machine learning algorithms with the CNN model and see which is better for prediction?

The structure of this paper is as follows, 2. Literature review, 3. Methodology, 4. Data processing and Scaling methods, 5. Ethical considerations and SWOT analysis, 6. Evaluation metrics and Validation, 7. Results, 8. Conclusion and Future plan, 8. References, and Appendix.

2. Literature Review

Image processing is a trending and interesting technology to work with. There are many experiments conducted on disease detection in plants based on image processing methods. This section includes the knowledge gained from different papers and the methodologies used by different people in the different approaches. This section is mainly divided into three sections. There are Tools required for the analysis and common problems, Machine learning models, and deep learning models.

2.1 Tools required and Common problems

Initially, there are many methods, tools, and algorithms for implementing image processing models. These topics are explained clearly and give a basic idea about image processing from the Neptune.ai website (Neetika Khandwlwal 2022). This explains the libraries and functions useful for analyzing the images. In this study, the pillow library will be used for image data extraction. There are some challenges in the usage of image processing also and the most common issues are the quality and realism of the images(Dufaux 2021) The technical issues in image processing include light spots, shadows, and colour contrast. There are so many other techniques like dimensionality reduction, and feature extraction that can be used to reduce the effects of these challenges. The model may fail to detect the outputs if these problems are not reduced, so these are crucial at the initial analysis stage.

2.2 Machine learning models

The authors of the paper (Sanjay et al. 2013), proposed a methodology for disease detection using image processing. The RGB format of the images is converted to the HSV format, and the green pixels of the images are masked. In the further process, the masked green pixels will be removed, and segments will be obtained from the data. In the final stage, the features will be computed using the algorithm. The problem with this approach is segmentation and image size. The reduction of the image size can be good for analysis, but the clarity of the image is reduced and the analysis is only performed on the affected area or the part of the image which shows the disease. The diseases like yellow leaf curl virus will make the leaf curl and the removal of green pixels may remove the entire leaf. So, this approach is not suitable for our research. Renuka Rajendra Kajale (Rajendra Kajale 2015) also proposed a similar approach and extended the work to the android system. The usage of this system is good for the leaves which shows the disease on the surface but in our approach healthy and yellow leaf curl images are also involved which are represented in green colour.

Sagar and Khule experimented on tomato disease detection using image processing (Vetal and R.S. 2017). The study uses data from four types of tomato diseases and for image smoothing, they used kurtosis and skewness filters. The RGB images are converted to HIS format and later performed various classification algorithms on the data. The results show that multi-class SVM is giving better results than

the other algorithms. This study didn't include the healthy leaf in the analysis. The major evaluation method used for the evaluation is accuracy. More metrics needed to be examined and compared while comparison between the models.

The paper (Patil and Pawar 2017)or tap here to enter te proposed a system that is based on the RGB to HIS vector and masking the green pixels. Later, the green pixels are treated based on the threshold and the data is clustered using the K-means algorithm and applying different algorithms to know the best model for the analysis. This model can be used to detect the disease, but we are using healthy leaves also for the detection. By thresholding the green pixels, we may get only a shadow or no image of the healthy leaf. So, this approach is not suitable for our analysis.

Aditi and Harjeet conducted a study to detect the disease on potato leaves using image processing and the SVM algorithm(Singh and Kaur 2021). In the pre-processing section, they used the grayscale data and performed a K-means algorithm to extract the features from the images. The results from the SVM algorithm show an accuracy of 95.99 percent in detecting the disease and the other metrics also show more than 90 percent of the results. This study includes data on two diseases and healthy leaves.

In the study (Madiwalar and Wyawahare 2017), built a model based on the GLCM (Gray Level Co-occurrence Matrix) and SVM algorithm. The results show that the GLCM method can extract the features of almost all the diseases except one. The SVM classifier can predict the diseases better than the minimum distance classifier.

The authors of the study (Kulkarni et al. 2021) build a model to detect the diseases of five plants and the model using a random forest algorithm. In pre-processing, two colour features grey level co-occurrence matrix and HSV conversion are used. Later, The metrics from the random forest algorithm show that the model achieved more than 87 percent accuracy for all the five plants. We will use the random forest methodology in our analysis.

The research conducted by the authors (Islam et al. 2017) shows that disease detection can be achieved using image segmentation and the multi-class SVM algorithm. The research is conducted on the potato leaves and the data is converted to LAB colour space before analysis. The SVM linear algorithm achieves more than 90 percent of accuracy in all the stages. The different metrics also show the reasonable outputs of the analysis.

According to Patil's paper (Sanjay Patil 2011), The disease severity is estimated using thresholding by triangle method on the histogram. The lesion region area is taken for analysis and to estimate the severity of the disease. This study is based on the reduction of pesticide usage and knowing the disease severity in the sugarcane plants. This process is complex and a bit hard to understand. We are using more than 5000 images in the research and estimating all the images using this method is time-consuming and more complex to implement.

The authors of the paper (Piyush Chaudhary et al. 2012), experimented to know which transform method is giving reasonable output. They compared YCbCr Colour Model, HIS Colour Model, and CIELAB Colour Model. They finally suggested that the CIELAB colour model is giving reasonable outputs and it is good for disease spot detection on plants. The visualizations show that the CIELAB method removes the noise from the image. This approach needs to be extended by using the ine learning classification algorithms on each model and seeing the predictions of each model.

According to the paper (Zhang et al. 2015), disease detection is performed on the segmentation images with the help of the K-nearest neighbours algorithm. The RGB image is converted into HSV format, and the green pixels are masked on the images. The visualizations show how the disease is detected from the images. The results show that the algorithm achieved more than 90 percent accuracy.

The authors of the paper (Jagtap et al. 2019) proposed an internet of things-based machine learning model which is useful for packing potatoes. This model uses the camera and captures the picture of a potato that is flowing on the belt and analyzes if there is any disease present on the potato based on the previous data. The mechanical arm will eliminate the diseased potato from packing based on the input from the image. This process involves a high-performance algorithm that is capable of detecting the disease in a few seconds and giving input to the mechanical arm.

In the study (Kurniawati et al. 2009), The image processing techniques for paddy disease detection are studied. They compared the metrics of two thresholding techniques using MATLAB. The results show that the Local entropy method is better than the OTSU method for image processing in paddy disease detection.

2.3 Deep Learning Models

Nagamani and sarojadevi(Nagamani H S and Dr. Sarojadevi H 2022) used deep learning algorithms to predict the disease on the tomato leaves. In the pre-processing stage, the images are converted to grayscale, and the size of the images is reduced. They implemented SVM, CNN, and R-CNN on the data and they suggest using R-CNN among the three models. They used the confusion matrix and performance chart for the evaluation and R-CNN shows good metrics in the results. They suggest the usage of deep learning models reduced the pre-processing process more than the traditional image processing models.

According to the authors (Singh and Misra 2017), performing an image smoothing filter and masking green pixels will enhance the computational efficiency of the model. They converted RGB images to HIS format and performed clustering using k-means and KNN. The experiment is conducted using MATLAB and they use data from different plants. They compared the accuracies of both K-mean and KNN for picking the better model. The results show that KNN is better for the classification than

KMEANS for almost all the plant leaves used in their study. They also suggest implementing deep learning algorithms like CNN and Bayes classifier for better results.

Sachin and Patil proposed various techniques used for image processing(Khirade and Patil 2015). In the pre-processing section, the RGB images are converted to grayscale images. Coming to the image segmentation section, they mentioned the different algorithms and they are boundary and spot detection algorithm, K-means clustering, and OTSU threshold algorithm. In the feature extraction section, they mentioned the colour co-occurrence method, HIS, and the LAB colour space method. For classification, the paper suggests using the ANN and BPNN algorithm. This paper is really helpful to know the basics and the popular methods in image processing.

According to the authors of the paper (Badnakhe and Deshmukh 2011), using the colour co-occurrence method(CCM) for feature extraction and performing the Neural network algorithms will give good results in the prediction of diseases. The proposed model suggests implementing an RGB vector and performing a K-means algorithm on the vector to cluster the data into different categories. In our experiment, the data has the response variable and there is no need to cluster the data. This approach is more suitable for semi-supervised models.

The authors of the paper (Mim et al. 2020), build a model to detect the diseases in the tomato leaves using the CNN algorithm. In the pre-processing stage, the images are resized the image size to 128×128. In the analysis stage, by changing the learning rate and running the CNN algorithm more times they achieved reasonable results. This approach uses the deep learning methodology, and the model is re-run until reasonable metrics are obtained.

The authors of the research (Rangarajan et al. 2018) on the pre-trained models to detect tomato disease shows that accuracy changes with the change in the dataset size. The study is conducted using two deep learning algorithms namely Alexnet and VGG16 net. The results show that both the algorithms performed well when the total dataset is taken and the accuracy changes with the decrease in dataset size. The metrics of both algorithms changes with the change in learning rate and batch size. The results show that the accuracy increases while using more data. We will build a model with complete data and reduced count data in our analysis to explore the effect of count on accuracy.

In the research (de Luna et al. 2019), They followed the AlexNet approach for building the model to predict the disease on the tomato leaves. They used the RCNN for fast processing of the data and to detect the disease in the tomato leaves. The images are connected to a network for better analysis and the user can monitor the health of the plant using an internet connection. This requires more hardware and resources for connectivity.

The study from the paper (Trivedi et al. 2021) suggests building a model based on the CNN algorithm will give good accuracy and performs well in the prediction of new images. The model gives good

metrics and performs well with all the metrics. The model performs well with the training and testing accuracies, and it outperforms when compared to the other models. This requires less pre-processing and it can be extendable to the other crops also.

According to the authors of the research (Sharma et al. 2020), CNN algorithm accuracy outraced the accuracy when compared with Logistic, KNN, and SVM. The study is conducted on data that is converted from the RGB format to the HSV format and it contains 20000 images and 19 classes. The final accuracy metrics show that the traditional machine learning algorithms achieve less accuracy and the CNN is best for large datasets.

The authors of the study (Salih et al. 2020) implemented a deep learning model on the tomato leaves to detect the disease. They resized the data and classified it according to the disease. Later, they performed CNN algorithm on the data with different dataset sizes and the maximum accuracy is obtained from the split dataset with 80 percent training data and 20 percent testing data.

2.4 Understanding from related works

From the related works, we can see that most of the works use four steps for the analysis. They are image pre-processing, image segmentation, Feature extraction, and Image classification.

1. Image pre-processing: most of them are using the traditional grayscale or RGB to HSV.
2. Image Segmentation: most of the works use K-means for the segmentation and others are using the disease.
3. Algorithms: The most used classification algorithms are SVM and KNN machine learning, and CNN while using deep learning models.

An image is a collection of different pixels, and the values of these pixels decide the colour of the pixel. Humans can classify the images by looking at and analyzing them but the computer needs the image in vector form for analysis and classification. The images must be converted to different vector formats for analyzing them using machine learning. The images are converted to different vector formats like RGB and grayscale. The individual channel information of each colour can be extracted from the RGB vector using the list functions. We can resize the images and extract the information to speed up the process. In this research, the data is studied in four different ways. Those are 32×32 dimension images data, 64×64 dimension images data, data with a maximum count of 600 in each class for 32×32 dimension images, data with a maximum count of 600 in each class for 64×64 dimension images, and 128×128 dimension images data. The original images are resized to increase the performance of the model. These four different types of analysis are explained using Table 1.

Data Type	Description
128×128	Data with 128×128-dimension images
Balanced data (32×32 and 64×64)	Data with a maximum count of 600 in each class is created for analysis
32×32	Data with 32×32-dimension images
64×64	Data with 64×64-dimension images

Table .1 Data used in this study

Knowledge about the different image vectors and classification algorithms is much needed before starting the project. If the pixels are high, then the vector data is large and takes more time to process the classification. So, it is better to resize the images to lower pixels and run the model. For checking more pixels can increase the accuracy, the dataset is reduced to 3 different pixel sizes. We use different parametric and non-parametric models for the classification of the images. As seen in the related works, the models are compared using different classification metrics. More details about the algorithms and the vectors will be discussed in the methodology section.

3. Methodology

Processing the data using different steps and placing them in the correct order is required for the representation of the results. A proper methodology will give a proper picture of the different stages of the research. This study follows the structure and process of the CRISP-DM methodology. CRISP-DM is an acronym for the Cross-industry standard process for data mining. CRISP-DM is a powerful and robust method for implementing research. This is an open-source methodology and is widely used by data practitioners for research. It consists of six major phases and the approach follows a step-by-step implementation of each phase(Ayele 2020). The phases in the CRISP-DM methodology are Context understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. The implementations and importance of each phase will be explained in the later sections of this paper. Other advantages of this methodology are flexibility and dependencies. Each phase has certain methods and rules which will impact the results. So, it is important to strictly follow the same pattern in the overall process. The researcher can easily move back to the sequence or restart the phase based on the research and the dependency of the result in each phase can be observed clearly. All these processes start after the data collection process because finding the correct data according to the context of the research is essential at the initial stage. By using this methodology, we can showcase the models of each dataset and represent the results clearly. The coding is saved in GitHub after the completion of some progress. The methodology of this project is explained in Figure 1.

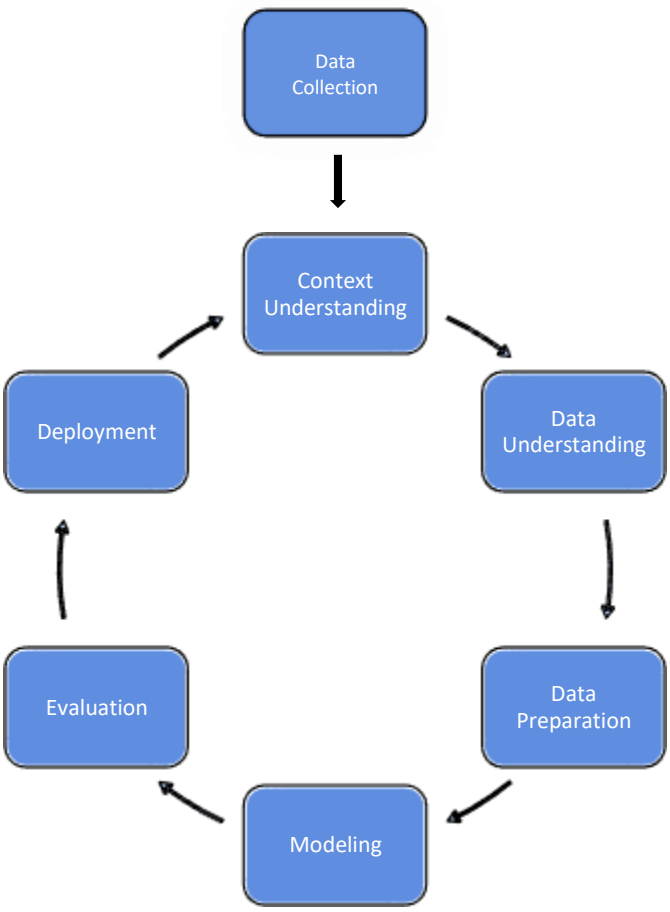


Figure 1. Methodology of the project

The GitHub link for coding of this research is: https://github.com/rcreddykovvuri/dissertation.git

3.1 Data Collection

In the data collection phase, the primary data needed for this research is the images. Collecting the images from the farmers or from a certain location is a hard and time taking process. So, the best way to collect data is by taking data from the internet and following the rules and regulations of data collection. There are some ethical and economic considerations for collecting data from the internet. The data should be from an open source with no restriction on usage, data should be relevant, and data should be hygienic. The data which follows all these principles should be taken for further analysis. Data relevance and data hygiene is also the most important factors because irrelevant or blurry images are not good for the analysis. The data must contain the healthy leaves images also. After searching and exploring different related works and datasets related to the tomato leaves images, there is a website named Mendeley(Huang and Chang 2020) which gives information about the tomato leaves. It consists of images about nine different diseases infecting the tomato leaves along with the healthy tomato leave images. The data is open-source, hygienic, and relevant to the tomato leaves. The images have a white background and look good for understanding. The ethical considerations of this data will be discussed in the later sections of this paper.

3.2 Context Understanding

The next phase of the research is to understand the context of the research. The context of the research is to help farmers by implementing a model which is capable of detecting diseases infecting tomato leaves using image processing and machine learning techniques. To achieve this context, research should be conducted on different vector types and images of different sizes. A programming language is needed to start the research and perform an analysis of the images. Python programming language is used for the overall analysis of the data because it is one of the most popular, robust language, and many libraries are available for data analysis. The major phases that need python libraries for this research are converting images to vectors, saving the data, and conducting the analysis by performing different algorithms. The Pillow library is used for reading the images and conversion of the images to vectors. For handling the data vectors and operations, the major libraries used are Pandas and NumPy. Finally, the sci-kit learn library is needed for analyzing the data using machine learning techniques and tensor flow is used for implementing the CNN models. There is no special equipment or permissions needed for proceeding with the analysis because the data is taken from an ethically significant source and the output of the models is shown on the computer. There is no personal information or human-related data used in this model.

3.3 Data Understanding

The data is separated according to the diseases and placed in different folders with the disease name as the folder name. All the images are in jpg format and the dimensions are 227×227. All the images are in colour format and there are no grayscale images. There is a total of ten folders among them, nine are related to diseases infecting tomato leaves and one folder contains images of healthy tomato leaves. The nine diseases are Bacterial spot, Early blight, Late blight, Leaf mold, Septoria leaf spot, Target spot, Tomato mosaic virus, Tomato yellow leaf curl virus, and Two-spotted spider mite.

The description of each disease symptoms on leaves is as follows. Bacterial spot is a disease that primarily infects the leaves and green tomatoes. The leaves of the tomato plants will get water-soaked spots on them. Early blight is a disease caused by a fungus called A. solani and this causes a spot with a bull's eye pattern in the leaf. The leaf tissue which is nearer to the affected area may turn yellow. Coming to the late blight disease, it will mainly cause during the cool and winter seasons. The symptoms show shriveling of leaves and the browning of leaves. Another dangerous disease is leaf mold which causes yellowish or pale green spots on the leaves and if untreated it will infect the stem also. Septoria leaf spot is a destructive disease, and it can be usually detected under the leaf. This reduces the quality of the tomatoes(Marjan Kluepfel et al. 2021).

Target spot is a disease caused by a fungus and it is misunderstood as early blight or bacterial spot in the early stages(Gene McAvoy 2020). This causes lesions on the leaf and damages the plant. Mosaic

virus is a devasting disease and the leaves become stunted when infected(Amy Grant 2021). Coning to the yellow leaf curl, it is an extremely damaging disease to the yield(Marjan Kluepfel et al. 2021). The major symptom is the curl of the leaf. Finally, the Two-spotted spider mite is a common disease and the major symptom is a webbing pattern on the leaf(R. Hazzard 2013). The sample images of all the diseases and healthy leaf are shown in Figure 2.

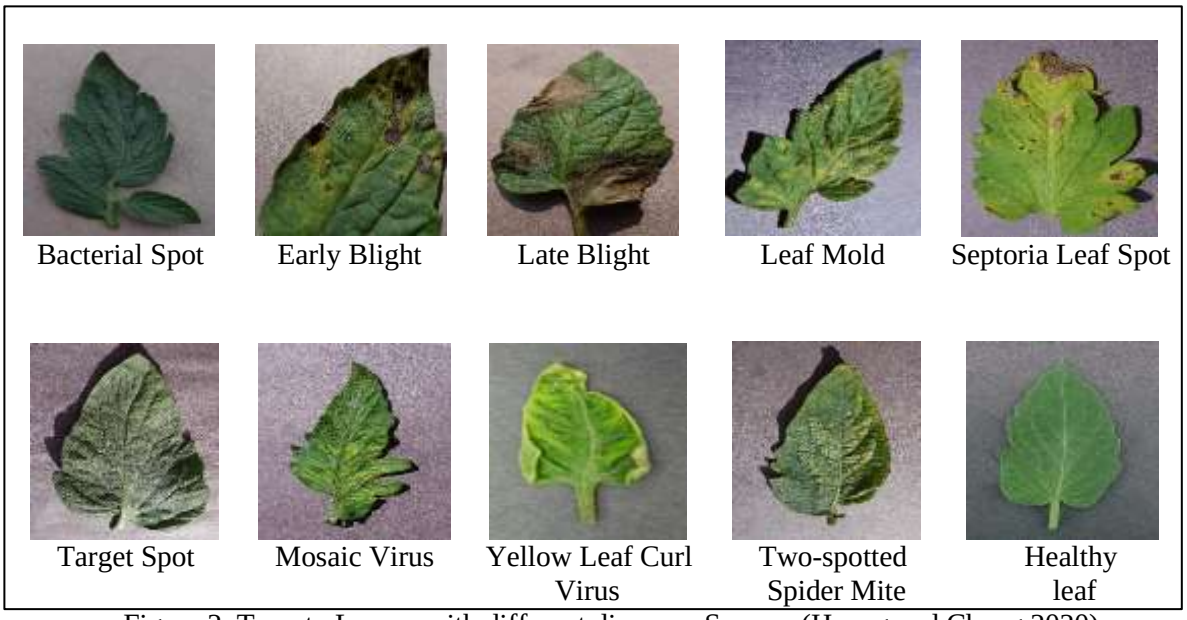


Figure 2. Tomato Leaves with different diseases. Source: (Huang and Chang 2020)

The count of data is not uniform in the dataset and there are certain classes with more data and some classes with fewer data. The highest number of images are available in the yellow leaf curl virus class and the least amount of data is available in the mosaic virus class. The count of images in each class is explained by figure 3.

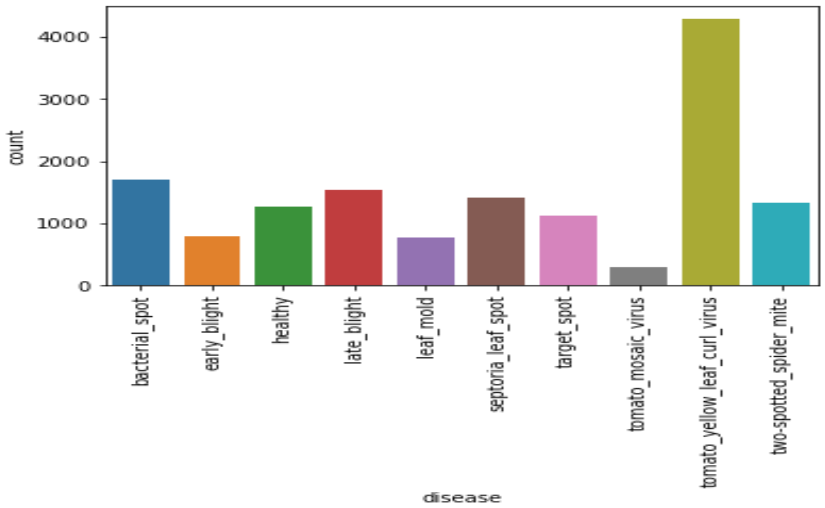


Figure 3. Count of images from each class from the dataset.

For analyzing the balanced dataset, The images are reduced in each class and the dataset is formed by taking a maximum count of 600 for each class. The only class with less than 600 images is the tomato

mosaic virus images. The remaining classes contain 600 images. The total count of images becomes 5699 in the balanced dataset. The count of images in each class of the balanced dataset is shown in figure 4.

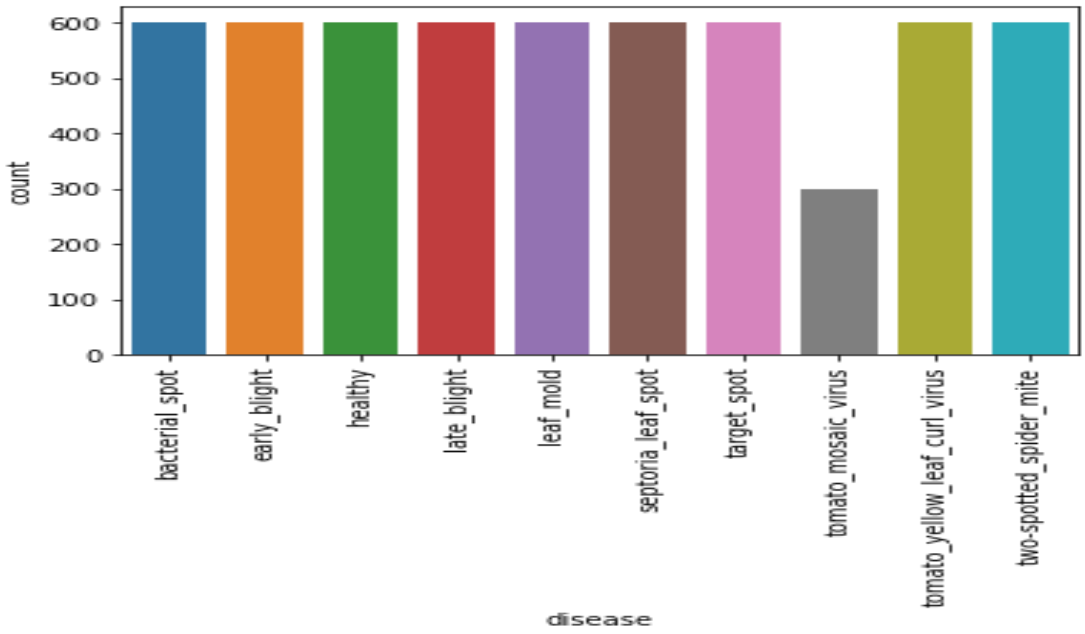


Figure 4. Count of images the in balanced dataset

The research is carried out on both unbalanced and balanced datasets to know the performance of each dataset with different classification models.

3.4 Data preparation

Image data needs to be changed to different vector forms for analysis. Image is made up of a combination of pixels and each pixel has an RGB value. The colour of the pixel is displayed based on the RGB value. For grayscale images, there is no RGB value but there is a single colour value for each pixel. In the preparation phase, The data is copied to different folders and resized the data based on the dimensions needed. The vector of each image is extracted using the pillow library function. These vector values of all pixels of each image are saved in the NumPy array because vector data takes more memory in the CSV files. For example, if we take a vector of a 32×32 dimension the number of pixels of the image is 1024 and each pixel contains 3 channels that make the flattened array length 3072 columns. This requires more memory if we take in the CSV file. So, the array of vectors is saved into a NumPy file and the other features like mean and disease name are saved in the CSV file. Firstly, The images are opened in the python program and the RGB vector of the image is taken into a list. Later, the image is converted to grayscale, and its vector value is saved to an array. The RGB vector is a three-dimensional array and the grayscale vector is a two-dimensional array. The grayscale vector is flattened to a one-dimensional array and saved to a NumPy file in the local system. Coming to the RGB vector,

the red channel vector data, the green channel vector data, and the blue channel vector data are extracted from the RGB vector and saved into different lists. The name of the disease and the mean of the vector are saved in a CSV file for all the formats. Finally, there will be two files for each vectorization method and the total number of files are 10. The same process will be repeated on the PCA dataset and balanced count dataset. The only difference for the balanced count dataset is, that the same process will be repeated by taking a maximum of 600 images of each disease in all three formats. The preparation of the data is explained more clearly in the data pre-processing section of this paper.

3.5 Modeling

This is multiclass data and can be classified using different classification algorithms. The data is modeled using different parametric and non-parametric algorithms to test which models are more appropriate for the analysis. The parametric models make strong assumptions about the data before fitting it and this will limit the learning of the data also. The parametric models are good with fewer data problems and they train the models faster than the non-parametric models. The problem with parametric models is learning the underlying pattern of the data, the chances of poor fit, and more suitable for simple problems. If the data pattern is not linear then it is unable to predict the data. On the other hand, non-parametric models do not make any strong assumptions about the data. Non-parametric models try to learn the functional form of the data freely. So, the major advantages of using non-parametric models are performance, power, and flexibility with complex problems. The disadvantages of non-parametric models are they require more training data, slower than the parametric models, and risk of overfitting the data(Jason Brownlee 2020). Both parametric and non-parametric models are applied to know whether the dataset is a simple or complex problem. The algorithms used for this multiclass classification problem are listed in Table 2.

Parametric Models	**Non-parametric**
Logistic Regression, SVM(kernel = Linear), Gaussian Naive Bayes, Bernoulli Naive Bayes, And Multinomial Naive Bayes.	SVM(Kernel = RBF), SVM(kernel = Poly), SVM(kernel = sigmoid), KNN, Decision Tree, and Random Forest.

Table 2. Algorithms used in this project

3.5.1 Parametric Models

Logistic Regression is a famous classification algorithm used in classification problems. Logistic regression is easy to interpret and efficient for 2 class binary models. Normally, logistic regression is used for the target variable with 0 or 1 and it uses binomial probability on the data. We can use the

logistic regression for multi-class classification by using the multinomial logistic regression model. The multinomial logistic regression uses multinomial probability. It uses the one vs rest technique for classifying multi-class problems. For example, assume the target as healthy_leaf in our data for an observation. Then the data is taken in a binary way that only the healthy_leaf class is 1 and the rest of all the classes are 0 for that observation. It uses a multinomial distribution function for multiclass classification problem and fit the data. In the logistic regression function 'lbfgs' multiclass solver is used for this data.

The support vector machine (SVM) algorithm is useful in regression, classification, and outlier detection. SVM algorithm performs well on the high dimensional datasets. SVM tries to separate the data using a hyperlane. It has different kernels for different datasets. The linear kernel is used as a parametric method among all the other kernels. The linear kernel tries to separate the data using a solid line. It assumes the data can be separated by a linear line. SVM is mostly useful when the dataset has more features. The linear kernel fails to predict well if the data cannot be separated by a linear line(Prateek Bajaj 2018).

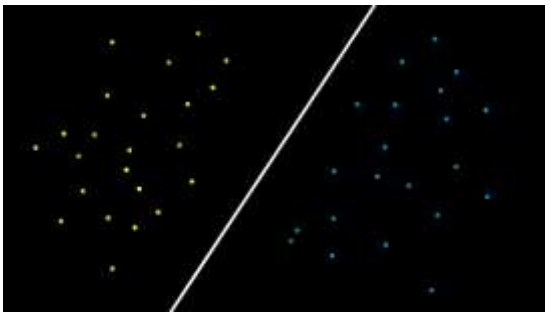


Figure 5. SVM separated two classes using a linear kernel. (source: (Prateek Bajaj 2018))

Naive Bayes algorithms are a set of parametric algorithms that assume the data based on the function. In this research, we use three naive Bayes algorithms in our research. Gaussian naive Bayes assumes the features are gaussian distributed. Multinomial Naive Bayes is used for text classification data and it uses multinomially distributed data. Coming to the Bernoulli naive bayes, it assumes the feature data is in the binary value. It will convert the input features to binary if we fit the model with non-binary data. The details about the naive bayes algorithms can be further explored in the sklearn documentation.

3.5.2 Non-parametric Models

As discussed earlier, the support vector machine algorithm can be used for both parametric and non-parametric models. The trick to finding the best SVM kernel is to try different kernels on the data. It will take the RBF kernel if we do not specify the kernel to the SVC function. RBF means radial basis function and it uses the gamma value for analysis. The gamma values is between 0 to 1. RBF is useful

when we do not have any idea about the data. It separates the data without the linear lines. Support vector machines with the poly kernel can separate the data using the decision boundary. It is useful when the data is distributed in a way separate by the boundary. SVM sigmoid is another kernel that is better for neural networks. The sigmoid model is like the two-layer perceptron model of the neural network. The different ways of separating the data are explained in figure 6.

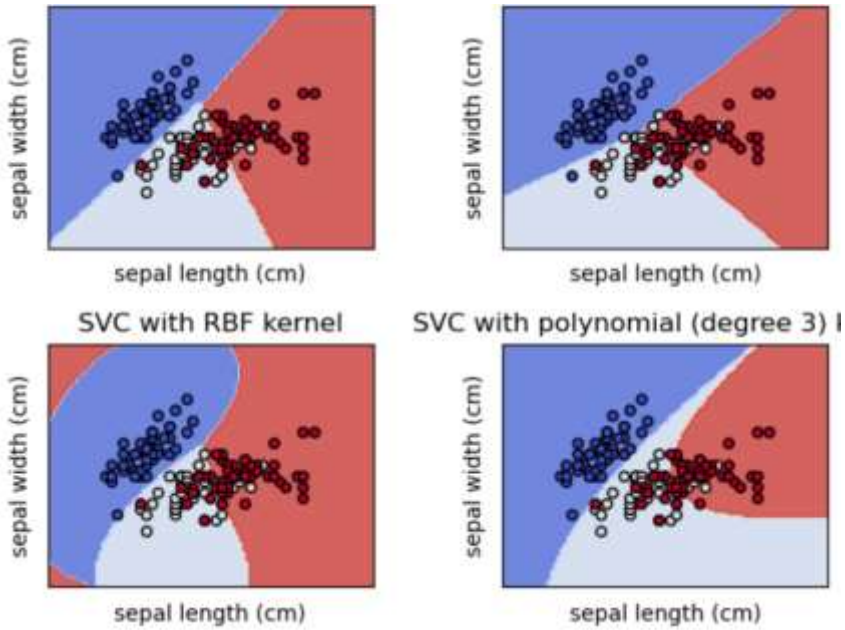


Figure 6. separation of data by different SVM kernels (source: SVM documentation in sklearn)

KNN algorithm decides the output of unknown data using its neighbours. KNN is a supervised learning classifier used to classify the data of the classes. Simply, it thinks that similar data always stay together. The number of neighbours that need to consider for computing the model is given by the k-value. The best way to choose the k value is by looking at the number of rows and taking root of the it. For example, the data has 9 rows. Then the preferable k-value is three for predicting the class of new data. It depends on the class of the three nearest data points near it. It is like a voting method by the nearest neighhbours on the new data. The disadvantages of KNN are, that it is a lazy algorithm and takes more time and memory, it performs badly with high-dimensional datasets, and there are chances of overfitting the model(What is the k-nearest neighbors algorithm? | IBM n.d.).

The decision tree is a supervised algorithm. The data is predicted using the decisions at every stage. Based on the decision at every stage, the final value is predicted. The root block of the tree is available at the top of the chart and the question of the decision starts at that point. The final decision stage where there is no chance to divide further is known as leaves(Amrutha K 2022). The decision tree is easy to understand and implement, able to handle multi-output problems, and has little preparation needed. The disadvantage of the decision tree is a small minor mistake may form a completely different tree and predict the data badly. The basic diagram of the decision tree is shown in figure 7.

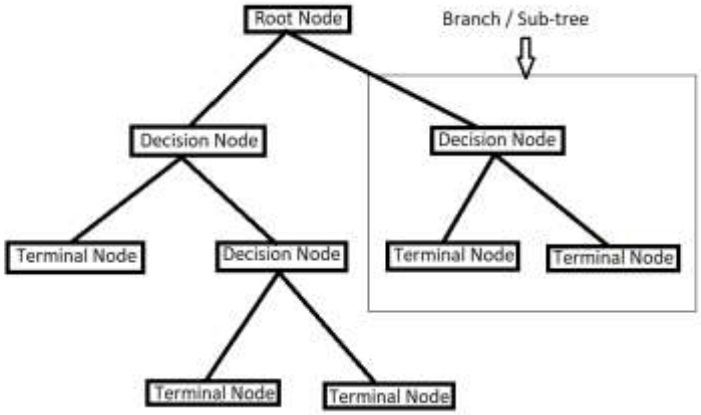


Figure 7. Basic decision tree structure example. (source: (Amrutha K 2022))

The random forest algorithm is a collection of different decision trees. So, it will take the decision based on the best votes of all the trees and provide the output. The random forest uses the wisdom of the crowd's principle. Each individual tree will give the output based on the spits and tell the class. The most votes of all the trees are taken as the output of the model. The default random forest algorithm takes the decision of 100 trees and predicts the output of the model(Tony Ylu 2019).

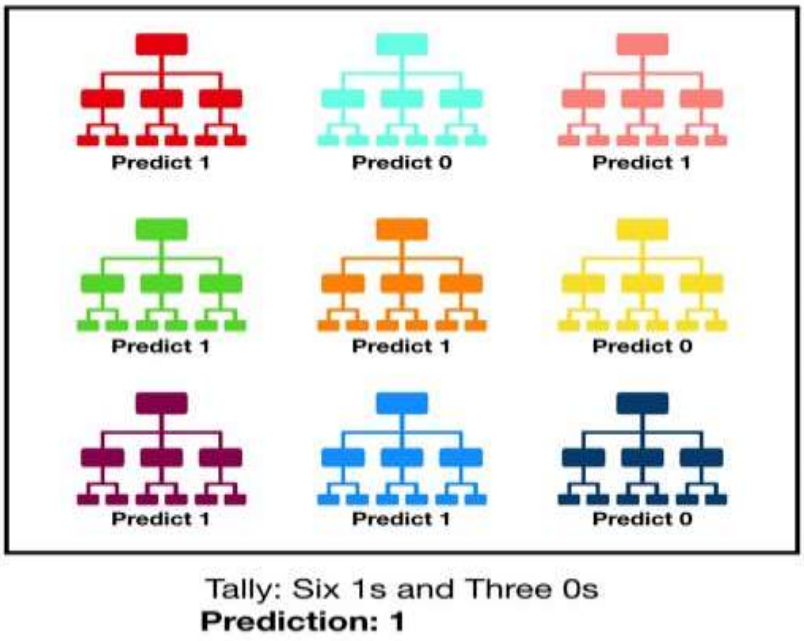


Figure 8. Basic random forest algorithm example. Source: (Tony Ylu 2019)

In the modeling phase, The converted data from the preparation phase is used to analyze the data by implementing different parametric and non-parametric machine learning algorithms. The process is to analyze the different data types separately. The array from the NumPy file is used as the features and the disease data in the CSV files will be used as the target variable. In the later stage, the data without doing any scaling and after doing feature scaling are analyzed. The scaling methods used in this research are standard scaling, Normalizer scaling, and Minmax scaling. All the datasets are examined using all

these scaling methods. Different scaling methods are used on the data and observe which method will be giving good prediction after performing algorithms. Since this is a multi-class classification problem, algorithms that are popular, used in related works and widely used for classification problems are performed on the data. From the available data, 80 percent of the data will be used for training and 20 percent of the data will be used for testing. In the first stage, the data will be analyzed without doing the feature reduction techniques and note the accuracy, precision, recall, and F1 score of the different models. Later PCA and ICA are applied to the best models to know the performance of the vector after applying feature reduction techniques. Only standard scaling is used in the PCA analysis.

3.6 Evaluation

Evaluation of the models is not an easy task. Looking at the accuracy alone will not give the best-performed model. The models will be analyzed using different metrics and the model which gives reasonable outputs in all the metrics will be moved to the deployment phase. The evaluation metrics that are considered in this research are accuracy, precision, confusion matrix, F1 score, recall, and classification report. In the first stage, the models with more than 80 percent accuracy will be taken. Later, the metrics of all the models will be examined to find the best model for the analysis. Apart from accuracy, the remaining evaluation methods will highly help in choosing the best model among the high accuracy models. If there is any problem with the results, then the process will be started again from the data preparation stage. That's why it is important to evaluate the models carefully throughout the examination process. There are many methods for each dataset in this research and the models are checked manually using the classification report of each model. After examining the models, the best scaling method, best vector method, and the best algorithm will be evaluated further. If two models are performing similarly, we can do a statistical paired t-test to compare the mean performance of the models and examine which model is performing well. To research more about the other features, the minimum, maximum, mean, and median are taken from each image along with the vector and trained in the model to examine if these features are increasing the accuracy of the model.

The model which shows good metrics is further examined using k-fold cross validation and AUC/ROC curve to validate and confirm the model is good at predicting the classes. K-fold cross-validation is one of the reliable techniques to analyze the performance of the model. K-fold uses different test and train data and shows the mean accuracy of the model in the end. The AUC/ROC curve will help to know which class is predicted well by the model using the probability of each class. By using these two techniques, we can say the final model is performing well on the data. The evaluation metrics used in this research are explained in detail in the evaluation metrics section.

3.7 Deployment

Finally, the model which shows good metrics and shows good performance in the validation is taken for the deployment. It is better to pick the best methods and train the method for deployment to ensure the accuracy of future predictions. The major goal of the deployment is to evaluate the context of the model and how the model can be deployed on the web. The model is saved as a pickle file to use in the future. There is no need to re-train the model for predicting while using the pickle file. We can directly load the model and start using it. After the deployment, we can examine how the model is working and how it is predicting the diseases of the tomato leaves. The deployed model can be used for the analysis as many times as possible. The pickle file can be deployed on the web also, but there is a need to implement the front end and maintain a small server for running the code. In this research, the final model is converted to a pickle file and tested using the python code.

4. Data processing and Scaling methods

The data processing stage is divided into two steps, the first step is implemented on all the datasets without doing any analysis and the second step is implemented after completing the analysis on the normal datasets and applying the elbow method to the best-performed datasets to get the number of components for PCA.

4.1. Step-1 Data processing at the initial stage

Images are understandable to humans and normally people will get more information from an image. This is not the case for analyzing images for machine learning. As mentioned before, Machines cannot understand the image directly. So, the data need to be converted to different vector forms for the analysis. Many Python libraries are available to extract the features from the image data. In this study, Pillow Library is used to convert the image data to vector form because it is a free and one of the widely used libraries to extract the features from images. Pillow library can do many functionalities on the images from basic operations like rotation and resizing to colour space conversions. It supports manipulations of many image formats, and it is easy to implement using python 3. Image data takes time to process, and it needs some powerful machines if there are more images. Another problem with image processing is time, image conversion takes more processing time, and it will slow down the overall process time. To counter these problems, we can save the vector formats and other information in the local system as NumPy files. Once the information is saved in the local system, we can use it for many purposes and in the future with reduced process time.

In this study, the data is saved in different sizes in separate folders. The data is converted to different vector forms and then data is saved in the local system using two formats for each vector form. The pixel data of different channels are saved in a separate NumPy array and the data about different features are stored in the CSV file. We can save the image vectors in a CSV file if there are images with fewer

pixels and fewer dimensions. We are dealing with more than five thousand images in all stages and there are images of 32×32, 64×64, and 128×128. If this data is saved in the CSV format, It will take more memory space and take a long time to process. Due to these problems, the CSV format is not a good pick for saving the image vector data in this case. The simple and efficient method of saving the vector data is by using the NumPy array file(npy). This will save the memory and we can read the data as an array easily. The main advantage of saving the data to vector form is memory saving and ease of use. We need a CSV file for saving the disease data to implement the classification of images and perform the machine learning algorithm on the data. The steps involved in the data processing are explained in Figure 5.

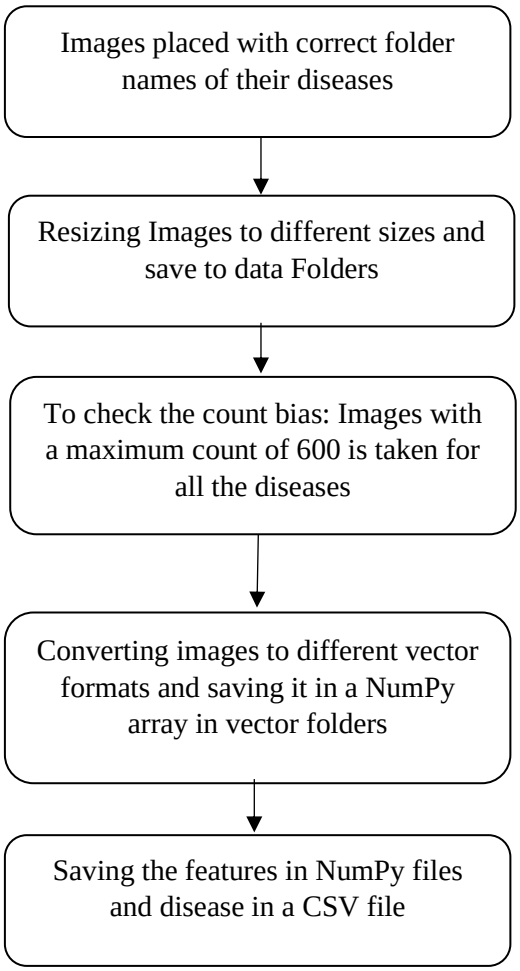


Figure 9. Data processing

The images are already separated using folders based on the disease type by the contributor which is available in the preprocessed_data folder. The name of the folder contains the disease name and some numbers. So, the first step in processing is to save the folders with the correct name. Later the images are resized to 32×32, 64×64, 128×128 sizes and saved in a separate folder. There are images with uneven counts, so images with a maximum count of 600 are taken from all the subfolders to reduce the bias in the count of the images.

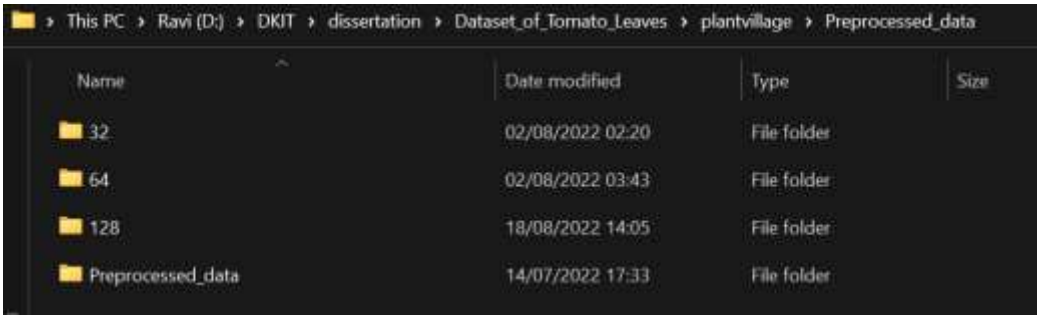


Figure 10. Data stored in different folders based on the dimensions

The images are converted to vectors and saved into a NumPy file, and the diseases are saved into a CSV file. By using the pillow library functions, the images are converted into Grayscale, RGB format, red channel, green channel, and blue channel vectors. The converted vector data from the images are in the array form and the data is flattened to the one-dimensional array before saving them. We can even use this data and reconstruct the image of each vector.

The below figures show how the images are converted to different vectors(Sandeep Balachandran 2020).

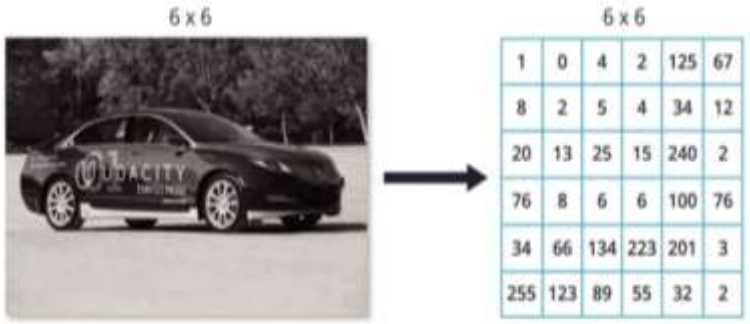


Figure 11. grayscale to vector. Source: (Sandeep Balachandran 2020)

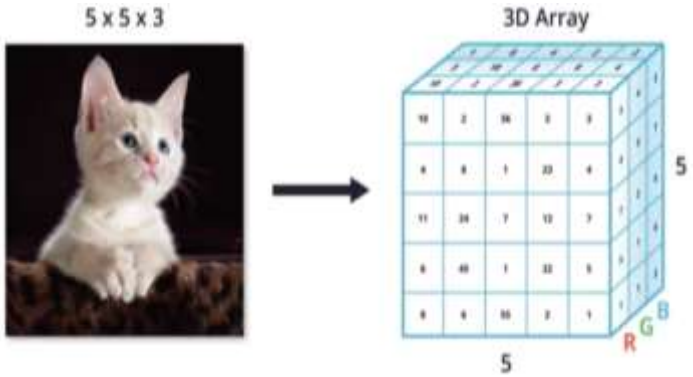


Figure 12. RGB image to vector format. Source: (Sandeep Balachandran 2020)

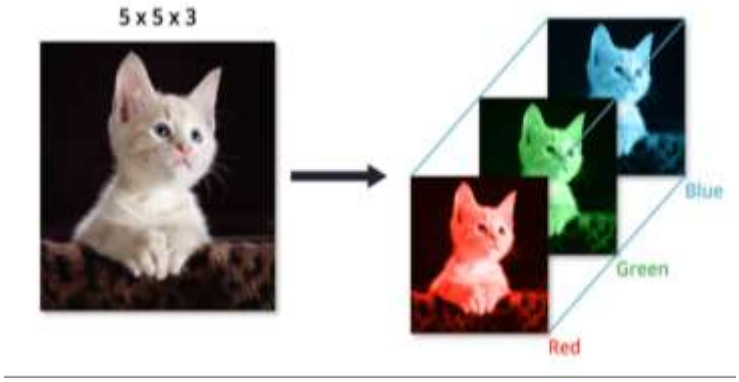


Figure 13. Image to RGB vector format. Source : (Sandeep Balachandran 2020)

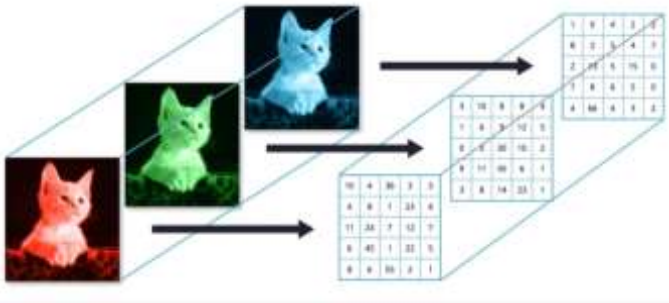


Figure 14. RGB to individual vector format. Source : (Sandeep Balachandran 2020)

For each category, different vector types of information are saved to different files individually. This information can be used for further analysis and to implement machine learning algorithms on the data. The below figure shows how the data is saved in the local system after taking vectors in the npy file and features in the CSV files.

blue_data.csv	31/05/2022 04:09	Microsoft Excel Com...	508 KB
blue_data.npy	31/05/2022 04:09	NPY File	58,125 KB
gray_data.csv	31/05/2022 04:09	Microsoft Excel Com...	513 KB
gray_test.npy	31/05/2022 04:09	NPY File	58,125 KB
green_data.csv	31/05/2022 04:09	Microsoft Excel Com...	513 KB
green_data.npy	31/05/2022 04:09	NPY File	58,125 KB
red_data.csv	31/05/2022 04:09	Microsoft Excel Com...	512 KB
red_data.npy	31/05/2022 04:09	NPY File	58,125 KB
rgb_data.csv	31/05/2022 04:09	Microsoft Excel Com...	540 KB
rgb_data.npy	31/05/2022 04:09	NPY File	174,373 KB

Figure 15. data of different vector forms

4.2. Step-2 Data processing after getting results from the initial analysis

After initial analysis of the datasets with different vectors, 32×32 and 64×64 image datasets gives noticeable results. So, the PCA analysis is carried out on both datasets only. The data is scaled first

using the standard scaling then principal component analysis(PCA) and independent component analysis(ICA) are carried out on the 32×32 images and 64×64 images. The number of components is selected based on the elbow method for all the vectors. The number of components needed to explain the variance is 5 for every vector and image size. The data is taken into the data frame and saved the data to a CSV file because the data only contains 5 components along with the disease name.

The below Table 2 shows how the data is saved in all the folders after processing the data into the folders based on the image dimensions:

Folder	Sub-Folders and data available
32	data(images with dimension 32×32), vector(npy and CSV files of images vectors for 32×32 dimension images), 32_no_bias(images with maximum count 600 of each disease), 32_no_bias_vector(.npy and CSV files of no_bias folder images), and PCA(CSV files of 32×32 images PCA data).
64	data(images with dimension 64×64), vector(npy and CSV files of images vectors for 64×64 dimension images), no_bias(images with maximum count 600 of each disease with 64x64), no_bias_vector(.npy and CSV files of no_bias folder images), and PCA(CSV files of 32×32 images PCA data).
128	data(images with dimension 128×128), vector(.npy and CSV files of images vectors for 128×128 dimension images),
Preprocessed_data	data(images from contributor), vector(.npy and CSV files of images in data folder).

Table 3. Data stored in the system after pre-processing

4.3 Scaling Methods

To know the effect of scaling on the data, the data is scaled using different scaling methods before splitting the dataset. The research is also performed without using any scaling method also to know whether scaling is important to this data or not. Generally, scaling plays a vital role in the output of the models. In some cases, the models which perform less accurately without scaling can perform well after doing the scaling. The scaling helps in getting good results based on the data. If the data have more outliers then they will dominate the analysis and the final model may fail to predict the values correctly. In these cases, scaling helps to organize the data according to the scaling method and helps in getting the patterns of the data. There are a plethora of scaling techniques and we can pick each scaling method to know which method is suitable for the data. In this research, three feature scaling techniques are used

to know the data. They are standard scaling, Minmax Scaling, and Normalizer. Each scaling technique will process the data differently. The scaling method of each scaler will be explained below.

4.3.1 Standard Scaling

Standard scaling uses mean and standard deviation to scale the data. So, after scaling the data will be centered to the mean and the standard deviation will be one. The values may be negative also based on the mean. The formula for standard scaling is as follows:

$$\text{x(new)} = \frac{x - x(mean)}{\sigma}$$

Where x is the original value, x(mean) is the mean of all the data, and σ is the standard deviation. We cannot use standard scaler data with the multinomial naïve Bayes model because it does not accept the data with negative numbers.

4.3.2 Minmax Scaling

Minmax scaler tries to fit the data between 0 and 1. The min-max scaling takes the maximum and minimum of the data column-wise and performs the scaling on the data. The formula for min-max scaling is as follows:

$$\text{X(new)} = \frac{x - X(min)}{x(\text{max}) - x(\text{min})}$$

Where x is the original value, x(min) is the minimum value of the column, and x(max) is the maximum value of the column.

4.3.3 Normalizer Scaling

Normalizer also fits the data between o and 1 like the minimax scaler. The only difference is that it takes the minimum and maximum values from the row, not from the column.

$$\text{X(new)} = \frac{x - X(min)}{x(\text{max}) - x(\text{min})}$$

Where x is the original value, x(min) is the minimum value of the row and x(max) is the maximum value of the row.

The research is performed by scaling the vector data with the scaler and fitting the data on different algorithms to know which scaling method is good for the tomato leaves dataset. Even though there are many scaling techniques available, the popular and widely used scaling methods are used in this research by choosing from related works. Coming to the CNN algorithm the data is normalized by dividing the features by 255, because the colour value of each pixel is present between 0 and 255.

5. Ethical Considerations and SWOT Analysis

5.1 Ethical Considerations

Ethical considerations are crucial and mandatory for any research. In practice, negligence of ethics will create a life and death situation. It is the responsibility of the researcher to evaluate the ethical considerations of a project before taking the data and following the ethics throughout the process. In this study, images of tomato leaves are used for disease prediction and image data also have many benefits and harms. The images used in this project are taken from a public website and except citation of the contributor, there are not many permissions needed for the usage of these images. The images are contributed by Huang and Chang. The information is available on the Mendeley data website(Huang and Chang 2020). It is free to use, and modifications of the data are also allowed. So, there is no ethical issue with the usage of the data but there are some ethically significant benefits and harms related to this data.

The major ethical benefits of using this image data are Human understanding, time-saving, predictive accuracy, and Economic efficiency. Any person can see the images used in this research and get a basic idea about the diseases. Different diseases affect the plant leaves differently and we can see the effects of each disease using these images. By using this model in real-time, we can reduce the time needed to wait for the agriculture professional and the farmer can start the implementation of the precautions in the early stage. Since this is a classification problem, the accuracy of the model is one of the major metrics and the percentage of accuracy is easily understandable to most people. If a farmer wants to re-examine the crops after some time, then he must again pay for the testing. This model can be built once, and we can use it as many times as required without spending more money. By implementing this model, we can reduce the cost of testing the crops.

The major harms with data are harm to privacy, harm to security harms to fairness and justice, and harms to transparency and autonomy. Images data will be predicted wrongly if there are colour spots or light fields on the images. This will cause the model to predict the data with less accuracy and this is harmful to the fairness and justice rules. The farmer needs to use pesticides based on the disease and the wrong prediction will destroy the crops. Overfit model may produce wrong outputs and using precautions without thinking may harm the crops. The researcher has to take care of these issues while building the model for detecting the to disease on the leaves. These are the major challenges with this data. There is no human-related information or unethical images are used in this analysis. The images of tomato leaves are used for the analysis and the credits are given to the contributor. So, there is no need to consider the privacy and security harms of the data.

5.2 SWOT Analysis:

SWOT is a strategy that organizes project-related ideas in a list. SWOT means Strengths, Weakness, Opportunities, and Threats. This helps in analyzing big problems systematically and more easily(Cecilia Lazzaro Blasbalg 2021). It is an initial step for properly organizing all the ideas and examine what are the internal and external points. Strengths and weaknesses come under internal origin and we have control over them. Weakness and opportunities are external origins, and we don't have control over them. By applying this strategy, there is a clear idea about the effectiveness and impacts of this project(Noah Parsons 2021).

5.2.1 Strengths

- Cost-effective: this can be implemented with less amount and there is no need to pay money for a medical professional for normal checking.
- Accuracy: Normally, image processing models give good accuracy and we can move to deep learning models if we have more data in the future. Each pixel value will help in finding the pattern and predicting the data.
- Time reduction: This will reduce the time taken for examining the disease. For example, a farmer can use this model if the farmer just wants to take a second opinion about the disease before using a pesticide.
- Flexible vectors: The vector information is stored in the form of npy. So, this data less time to process the information. So, we can easily add the information to the training data if we have more data in the future.

5.2.2 Weakness

- Time-consuming: Takes more time for converting the data to vector form. The initial data extraction steps take time for making the data. The training time of the data increases based on the dimensions.
- Noise Effect: If there is noise in the images then it is not capable to predict correctly. The input images should be taken with a white background to better predict the model.
- Reach financials: This model is built with the intention to help farmers. So, this needs financials to educate farmers about this model. All the farmers may not have the access to a computer, and they do not know how to use a computer. A minimum knowledge of computer skills is needed to check the results.

5.2.3 Opportunities

- Extendable: This model can be used for other crops also. We can use the same coding to build the model for other crops also and we can check the validation of the models based on the metrics.

- Geographical: More than 150 countries farm tomatoes and they can be used in all these countries. There is a need to consider the ethical permissions of each country before deploying the model.
- Partnership: Fertilizer companies invest in these types of technologies. We can collaborate with them for suggesting pesticides based on the disease. This can be executed according to the company's location and services.
- Trending technology: Image processing is a reliable and attractive technology. This is a reliable option for the future market. Almost many companies use image processing for different purposes for example, google lens uses images to predict the image data and show related recommendations.

5.2.4 Threats

- New Technologies: Deep Learning provides more accurate predictions if more data is provided. Machine learning methods are capable to handle the data up to a limit, but later we need to move to the deep learning models for getting good predictions.
- Services: Requires lots of services according to the crop and region.

6. Evaluation Metrics and Validation

The performance of the model can be calculated by using different evaluation methods. This is a classification model, and it needs to be evaluated using the classification metrics. The models are mainly evaluated using the classification report and the confusion matrix. The step-by-step detailed plan to evaluate the results is given in figure 16.

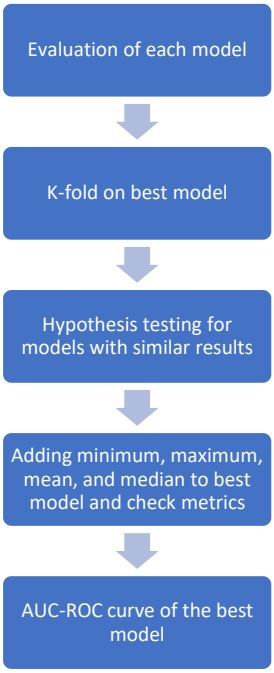


Figure 16. Evaluation and validation of the models

6.1 Evaluation of each model

Evaluation is a crucial step to select the best model from a plethora of models. This study is based on image processing and the machine learning classification process. The images from the contributor are clear and they don't need any filters. The size of the images only needs to be changed and there are no other changes needed. This study is based on the detection of leaves health with ten categories, so this is a multi-classification problem. For classification problems, most people evaluate the accuracy and move forward to the deployment, but it is not a good approach. In this study, different metrics will be examined after building a model. They are accuracy, precision, confusion matrix, F1 score, recall, and classification report. The evaluation metrics formulae are as follows:

1. Accuracy $= \frac{True\ Positive + True\ Negatives}{True\ Positive + True\ Negatives + False\ Postives + False\ Negatives}$

2. Precision $= \frac{True\ Postives}{True\ Positives + False\ Positives}$

3. Recall $= \frac{True\ positives}{True\ Positives + False\ Negatives}$

4. F1 Score $= \frac{Precision \times Recall}{Precision + Recall}$

Accuracy tells how good a model is to predict the output. The models are examined according to the accuracy first. Later, the models which give more than 80 percent accuracy will be selected and moved to further analysis. In the next step, The other metrics of each model will be examined manually using the classification report of each model. Generally, a classification report is a good method for evaluating multi-classification problems. The models which give a good score in the classification report and confusion matrix will be moved to further analysis.

6.2 K-fold Cross Validation

K-fold cross-validation is a good method to know the skill of a model and it reduces the bias in the decision while choosing the algorithm. The reason for choosing K-fold as validation for this research is that it takes different test data, and it divides the data randomly. K-fold takes the data as input and divides the data based on the number of folds given. The K represents the number of folds. The process of the K-fold cross-validation is shuffling the data, splitting the data, training and testing the data, and comparing the accuracy of the model with different folds. The main input of the k-fold function is the K value because based on the k value the data is split and the model is tested k times. We can take the

mean of the accuracy scores and compare them with our original model to know the skill of the model in predicting the data. For example, if the k-value is 5 then the data is shuffled to 5 folds equally. Then the data is tested 5 times by taking four folds for training and one-fold for testing. If the model is able to predict all the folds with nearer accuracies, then we can say the model is good for predicting the data. If the model is giving very low accuracies while using the k-fold then we have to verify the model again for the reason of low predicting. There are functions available to do the k-fold in python. In this research, cross-validation is carried out by using the sci-kit learn library. We can use the K-Fold function to build the K-fold cross-validation model and the cross_val_score() function to know the scores of the models with different folds. The output of the cross_val_score function is a list of the accuracies of each method. We can determine whether the model is good or bad by taking the mean and analyzing the accuracies. The only problem with the K-fold is processing time. It will take more time than the normal model because it runs for k times. K-fold is good for checking whether a single model is good or bad based on the accuracies, but for checking two models we have to do the cross-validation on two models and followed by a paired statistical test.

6.3 Hypothesis testing

There are cases where two models will give similar results and we cannot decide which model to choose based on accuracy, precision, recall, and F1-score metrics. In these cases, we can use a statistical test to know which model is performing well on the data(Jason Brownlee 2020a). The statistical test we used in this research is five repeats of two-fold cross-validation with a paired student's t-test. This statistical test is dependent because the same data is used. The statistical test will divide the data into 2 parts as training and testing, then it performs cross-validation on the 2 datasets for 5 times with each model using the same data. The accuracies and standard deviation of the results are taken for doing the statistical t-test of the two models. The null hypothesis of the statistical test is that "There is no difference between the mean performance of the model" and the alternate hypothesis is "There is a difference between the mean performance of the models". The hypothesis is accepted or rejected based on the p-value of the test. The alpha value taken for this test is 0.05 and if the p-value is greater than the alpha we can conclude that we failed to reject the null hypothesis that there is no difference in the mean performance of the models, and we can choose a model which gives good accuracy. If we got the p-value less than the alpha, then we have to choose the model that gives more accuracy of all means. We can examine the means manually or the best method to compare the model's accuracy mean score is using by using plots. We can use either a box plot or a whisker plot for analysing which model has good accuracy mean among the two models. This process is better than comparing the cross-validation scores of two models individually and getting the best model because we are performing the test by using the same data on both models.

The main advantage of using python is there are already libraries and functions for doing these tasks. This statistical test is also available in libraries that are already designed for this type of purpose. The whole discussed statistical test process can be easily performed using the mlxtend python library. The mlxtend library is designed by Sebastian Raschka and it is an open-source license library. So, there are no ethical issues using this library in this project. Firstly, we need to declare both algorithms we want to compare. The function used for performing the analysis is 'paired_ttest_5x2cv()'. The major input of this function is the 2 models we want to compare and the data. We have to give x as the feature data and y as the target data inside the function. There is no need to split the data because it will do the cross-validation on the data provided on both models. The output of the function is the t-statistic value and p-value. So, finally based on the p-value of this function we can decide which model is good for the data between the two models.

6.4 Adding Mean, Minimum, Maximum, and Median

In this research, new features are added to test the hypothesis that adding new features to the vectors and training them may increase the accuracy of the model. The research is applied to the best model which is obtained from the metrics and tests. The research is carried out on the best scaling method, and best algorithm but the only difference is adding the features before splitting the dataset. The new features are the mean, minimum, maximum, and median of each image. The extracted new features are stored in the CSV file along with the target variable and added to the features before splitting the data.

	mean	min	max	median	disease
0	97.180664	10	163	101.5	bacterial_spot
1	98.775716	5	184	86.0	bacterial_spot
2	115.811035	0	219	119.0	bacterial_spot
3	99.725260	11	192	89.0	bacterial_spot
4	104.349609	23	190	99.0	bacterial_spot
...	...	...	...	...	...
14526	128.490967	0	197	140.0	two-spotted_spider_mite
14527	113.974609	2	179	124.0	two-spotted_spider_mite
14528	124.632568	0	200	129.0	two-spotted_spider_mite
14529	117.917074	0	182	123.0	two-spotted_spider_mite
14530	123.845215	0	209	126.0	two-spotted_spider_mite

Figure 17. Sample of new features for images in a data frame

As shown in figure 17 the mean, minimum, maximum, and median of the data are read from the CSV file and assigned to a pandas data frame. These are added to the existing features data frame as columns using the pandas assigning. After adding These features then the data is scaled using the best scaling method. The data is then split into training data as 80 percent and testing data as 20 percent with the same random state used on the best model. Later, the data is fitted to the best algorithm and the classification report and confusion matrix of both the models will be compared. If the metrics are almost the same, then there is no change in the accuracy of the model with adding the features. If there is a drop in accuracy, then we can know the values are not advantageous for the model. The whole process is performed to know the extraction of new features from already existing features for this image data may be good for prediction or not. The statistical test is not useful because the features are different, and we are applying the same algorithm. So, the classification metrics such as accuracy, precision, recall, and F1-score are compared between the two models.

6.5 AUC–ROC curve

To validate and evaluate the model furthermore, we can use the AUC-ROC curve to the best model for knowing if the model is good or bad. The term AUC-ROC curve means "Area under the curve of Receiving operator characteristic". This helps us to validate the model using the graph. The AUC-ROC curve is a probability curve between the true positive rate against the False positive rate(Aniruddha Bhandari 2022). The formula for determining the true positive rate and the false positive rate is as follows:

$$\text{True positive rate} = \frac{\text{True positives}}{\text{True positives+False Negatives}}$$

True positive rate is also called recall and sensitivity.

$$\text{True Negative rate} = \frac{True\ Negatives}{True\ Negatives+False\ positives}$$

True negative rate is also called specificity.

$$\text{False Positive rate} = 1 - \text{specificity}$$

This is used to show how good a model is to predict each class in the data. ROC is used as a metric for binary problems to evaluate how well the model predicts the output. Here we calculate the probability of each class for each observation instead of just looking at the actual vs predicted value of observations. For example, if we have 3 classes and we have targets as 0,1, and 2 then the actual value is 0 and predicted as 0. Now if we examined the probability of this case, we can see the probability of class 0 is more than the other classes. So, we can say that the probability of class 0 maybe 0.6, class 1 is 0.20, and class 2 is 0.2. So, based on the probability there is a high chance that observation belongs to class

0. The ROC plot shows which class is better classified by the model. The area which is under the ROC curve is called the area under the curve(AUC). In simple terms, if the AUC value is 1 then the model is the best classifier but it is an ideal case. So, if the AUC value is nearer to 1 then we can say that the model is predicting the class well. If the AUC is less than 0.5 then we can say that the model is not good at classifying the class. Coming to the ROC, if the curve is plotted nearer to the y- axis then we can say the model is good at predicting the particular class. Normally, AUC-ROC needs a binary target variable but in our research there are 10 classes. Sci-kit learns library provides inbuilt functions for calculating the values. Firstly, The target data will be converted to binary data before doing the analysis. The probability parameter must be added to the machine learning algorithm before fitting. So we have 10 columns of probability for each image and the data consists of the probability of the image to belongs to a certain class. Later, we can calculate the ROC and AUC of each class using the sci-kit learn function. Finally, we can plot the data using the line plot to see which class is classified better by the model.

7. Results

In this section, the results from each section and each vector are placed to compare the best model for predicting the diseases. The results of each vector with no scaling, Standard Scaling, Minmax scaling, Normalizer, and PCA. The structure of the results section is explained by using Table 4.

Section	Image dimensions	Data Type	Vectors
7.1	32 × 32	Without Scaling, Standard Scaling, Minmax scaling, Normalizer scaling, and PCA	RGB, Grayscale, Red channel, Green channel, and Blue channel.
7.2	64 × 64	Without Scaling, Standard Scaling, Minmax scaling, Normalizer scaling, and PCA	RGB, Grayscale, Red channel, Green channel, and Blue channel.
7.3	Balanced data (32 × 32)	Without Scaling, Standard Scaling, Minmax scaling, Normalizer scaling	RGB, Grayscale, Red channel, Green channel, and Blue channel.
7.4	Balanced data (64 × 64)	Without Scaling, Standard Scaling, Minmax scaling, Normalizer scaling	RGB, Grayscale, Red channel, Green channel, and Blue channel.
7.5	128 × 128	Standard Scaling	RGB
7.6	CNN algorithm	Normalized data	RGB and Grayscale
7.7	Comparing the best machine learning models and CNN	Best scaling method from Machine learning	Best Vector from the validation section

7.8	Validation of the machine learning models	Best scaling method from Machine learning Metrics	Best vector based on the metrics
7.9	Adding more features to the best model and checking the classification report	The best vector from the validation	The best vector from the validation

Table 4. Results section structure

The weighted average of the metrics is taken into consideration for the model selection because of the imbalanced count of the classes in the data. The best model from the metrics will be compared with the deep learning model at the end of the results section. The initial research is carried out on all the datasets and the best scaling method is applied to the 128×128 dataset to validate the output.

7.1 Results of 32 × 32 dimensions images

7.1.1 Without scaling:

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.55	0.58	0.55	0.53
Logistic	0.63	0.63	0.63	0.63
SVM(linear)	0.73	0.73	0.73	0.73
SVM(poly)	0.80	0.75	0.75	0.75
SVM(RBF)	0.84	0.80	0.78	0.79
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.57	0.57	0.57	0.51
Gaussian Naive Bayes	0.51	0.58	0.51	0.52
Decision Tree	0.50	0.50	0.50	0.50
Bernoulli Naive Bayes	0.36	0.45	0.36	0.34
Multinomial Naive Bayes	0.36	0.45	0.36	0.34

Table 5. Results of 32 × 32 dimension images for RGB vector without scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.55	0.58	0.55	0.53
Logistic	0.63	0.63	0.63	0.63
SVM(linear)	0.73	0.73	0.73	0.73
SVM(poly)	0.58	0.59	0.58	0.58
SVM(RBF)	0.69	0.68	0.69	0.68
SVM(sigmoid)	0.24	0.09	0.24	0.13
Random Forest	0.57	0.57	0.57	0.51
Gaussian Naive Bayes	0.51	0.58	0.51	0.52
Decision Tree	0.50	0.51	0.50	0.51
Bernoulli Naive Bayes	0.31	0.32	0.31	0.16
Multinomial Naive Bayes	0.24	0.32	0.24	0.23

Table 6. Results of 32 × 32 dimension images for Grayscale vector without scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.55	0.58	0.55	0.53
Logistic	0.63	0.63	0.63	0.63
SVM(linear)	0.73	0.73	0.73	0.73
SVM(poly)	0.59	0.60	0.59	0.59
SVM(RBF)	0.69	0.68	0.69	0.67
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.57	0.57	0.57	0.51
Gaussian Naive Bayes	0.51	0.58	0.51	0.52
Decision Tree	0.49	0.50	0.49	0.49
Bernoulli Naive Bayes	0.31	0.26	0.31	0.19
Multinomial Naive Bayes	0.28	0.35	0.28	0.27

Table 7. Results of 32 × 32 dimension images for Red channel vector without scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.55	0.58	0.55	0.53
Logistic	0.63	0.63	0.63	0.63
SVM(linear)	0.73	0.73	0.73	0.73
SVM(poly)	0.58	0.59	0.58	0.58
SVM(RBF)	0.70	0.69	0.70	0.69
SVM(sigmoid)	0.27	0.09	0.27	0.13
Random Forest	0.57	0.57	0.57	0.51
Gaussian Naive Bayes	0.51	0.58	0.51	0.52
Decision Tree	0.50	0.51	0.50	0.50
Bernoulli Naive Bayes	0.32	0.24	0.32	0.18
Multinomial Naive Bayes	0.22	0.33	0.22	0.22

Table 8. Results of 32 × 32 dimension images for green channel vector without scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.55	0.58	0.55	0.53
Logistic	0.63	0.63	0.63	0.63
SVM(linear)	0.73	0.73	0.73	0.73
SVM(poly)	0.64	0.64	0.64	0.63
SVM(RBF)	0.72	0.71	0.72	0.71
SVM(sigmoid)	0.23	0.12	0.23	0.12
Random Forest	0.57	0.57	0.57	0.51
Gaussian Naive Bayes	0.51	0.58	0.51	0.52
Decision Tree	0.50	0.51	0.50	0.51
Bernoulli Naive Bayes	0.35	0.40	0.35	0.33
Multinomial Naive Bayes	0.43	0.49	0.43	0.44

Table 9. Results of 32 × 32 dimension images for Blue channel vector without scaling.

7.1.2 Standard Scaling:

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.59	0.61	0.59	0.57
Logistic	0.68	0.69	0.68	0.68
SVM(linear)	0.73	0.74	0.73	0.73
SVM(poly)	0.74	0.74	0.74	0.72
SVM(RBF)	0.85	0.84	0.85	0.84
SVM(sigmoid)	0.55	0.58	0.55	0.54
Random Forest	0.57	0.57	0.57	0.51
Gaussian Naive Bayes	0.51	0.58	0.51	0.52
Decision Tree	0.50	0.51	0.50	0.51
Bernoulli Naive bayes	0.43	0.52	0.43	0.43

Table 10. Results of 32 × 32 dimension images for RGB vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.49	0.50	0.49	0.46
Logistic	0.44	0.43	0.44	0.43
SVM(linear)	0.42	0.42	0.42	0.42
SVM(poly)	0.55	0.58	0.55	0.52
SVM(RBF)	0.70	0.70	0.70	0.69
SVM(sigmoid)	0.31	0.35	0.31	0.30
Random Forest	0.45	0.53	0.45	0.38
Gaussian Naive Bayes	0.37	0.45	0.37	0.34
Decision Tree	0.42	0.43	0.42	0.43
Bernoulli Naive Bayes	0.32	0.37	0.32	0.29

Table 11. Results of 32 × 32 dimension images for grayscale vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.51	0.50	0.51	0.48
Logistic	0.44	0.44	0.44	0.44
SVM(linear)	0.44	0.45	0.44	0.45
SVM(poly)	0.59	0.61	0.59	0.56
SVM(RBF)	0.70	0.69	0.70	0.69
SVM(sigmoid)	0.35	0.37	0.35	0.34
Random Forest	0.48	0.53	0.48	0.42
Gaussian Naive Bayes	0.38	0.46	0.38	0.36
Decision Tree	0.33	0.39	0.33	0.30
Bernoulli Naive Bayes	0.43	0.43	0.43	0.43

Table 12. Results of 32 × 32 dimension images for Red channel vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.48	0.50	0.48	0.45
Logistic	0.45	0.43	0.45	0.44
SVM(linear)	0.43	0.43	0.43	0.43
SVM(poly)	0.53	0.57	0.53	0.50
SVM(RBF)	0.71	0.70	0.71	0.70
SVM(sigmoid)	0.29	0.31	0.29	0.28
Random Forest	0.43	0.42	0.43	0.35
Gaussian Naive Bayes	0.35	0.42	0.35	0.32
Decision Tree	0.41	0.42	0.41	0.41
Bernoulli Naive Bayes	0.28	0.30	0.28	0.24

Table 13. Results of 32 × 32 dimension images for green channel vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.49	0.50	0.49	0.46
Logistic	0.44	0.43	0.44	0.43
SVM(linear)	0.42	0.42	0.42	0.42
SVM(poly)	0.55	0.58	0.55	0.52
SVM(RBF)	0.70	0.70	0.70	0.69
SVM(sigmoid)	0.31	0.35	0.31	0.30
Random Forest	0.45	0.53	0.45	0.38
Gaussian Naive Bayes	0.37	0.45	0.37	0.34
Decision Tree	0.42	0.42	0.42	0.42
Bernoulli Naive Bayes	0.32	0.37	0.32	0.29

Table 14. Results of 32 × 32 dimension images for blue channel vector with Standard scaling.

7.1.3 Minmax Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.58	0.60	0.58	0.56
Logistic	0.74	0.75	0.74	0.74
SVM(linear)	0.74	0.75	0.74	0.74
SVM(poly)	0.79	0.80	0.79	0.79
SVM(RBF)	0.83	0.83	0.83	0.83
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.57	0.57	0.57	0.52
Gaussian Naive Bayes	0.49	0.58	0.49	0.51
Decision Tree	0.46	0.49	0.46	0.47
Bernoulli Naive Bayes	0.36	0.45	0.36	0.34
Multinomial Naive Bayes	0.46	0.54	0.46	0.48

Table 15. Results of 32 × 32 dimension images for RGB vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.45	0.45	0.45	0.42
Logistic	0.48	0.46	0.48	0.47
SVM(linear)	0.49	0.48	0.49	0.48
SVM(poly)	0.55	0.57	0.55	0.55
SVM(RBF)	0.69	0.68	0.69	0.67
SVM(sigmoid)	0.23	0.09	0.23	0.13
Random Forest	0.45	0.53	0.45	0.38
Gaussian Naive Bayes	0.36	0.45	0.36	0.34
Decision Tree	0.38	0.38	0.38	0.38
Bernoulli Naive Bayes	0.31	0.31	0.31	0.17
Multinomial Naive Bayes	0.30	0.33	0.30	0.28

Table 16. Results of 32 × 32 dimension images for Grayscale vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.48	0.47	0.48	0.45
Logistic	0.49	0.48	0.49	0.48
SVM(linear)	0.51	0.50	0.51	0.50
SVM(poly)	0.57	0.59	0.57	0.58
SVM(RBF)	0.69	0.69	0.69	0.67
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.48	0.51	0.48	0.42
Gaussian Naive Bayes	0.37	0.47	0.37	0.35
Decision Tree	0.41	0.42	0.41	0.41
Bernoulli Naive Bayes	0.31	0.26	0.31	0.19
Multinomial Naive Bayes	0.35	0.38	0.35	0.34

Table 17. Results of 32 × 32 dimension images for Red channel vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.45	0.44	0.45	0.42
Logistic	0.48	0.46	0.48	0.46
SVM(linear)	0.50	0.48	0.50	0.48
SVM(poly)	0.54	0.58	0.54	0.55
SVM(RBF)	0.70	0.69	0.70	0.69
SVM(sigmoid)	0.27	0.09	0.27	0.13
Random Forest	0.42	0.43	0.42	0.34
Gaussian Naive Bayes	0.33	0.40	0.33	0.31
Decision Tree	0.37	0.38	0.37	0.37
Bernoulli Naive Bayes	0.32	0.25	0.32	0.18
Multinomial Naive Bayes	0.27	0.28	0.27	0.25

Table 18. Results of 32 × 32 dimension images for Green channel vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.54	0.58	0.54	0.53
Logistic	0.54	0.55	0.54	0.53
SVM(linear)	0.54	0.55	0.54	0.54
SVM(poly)	0.63	0.64	0.63	0.63
SVM(RBF)	0.72	0.72	0.72	0.71
SVM(sigmoid)	0.25	0.11	0.25	0.13
Random Forest	0.54	0.50	0.54	0.49
Gaussian Naive Bayes	0.50	0.57	0.50	0.52
Decision Tree	0.41	0.44	0.41	0.42
Bernoulli NaiveBayes	0.35	0.40	0.35	0.33
Multinomial Naive Bayes	0.46	0.47	0.46	0.45

Table 19. Results of 32 × 32 dimension images for blue channel vector with Minmax scaling.

7.1.4 Normalizer Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.58	0.59	0.58	0.55
Logistic	0.64	0.62	0.64	0.60
SVM(linear)	0.68	0.66	0.68	0.65
SVM(poly)	0.81	0.81	0.81	0.80
SVM(RBF)	0.83	0.82	0.83	0.82
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.53	0.55	0.53	0.44
Gaussian Naive Bayes	0.48	0.57	0.48	0.49
Decision Tree	0.48	0.49	0.48	0.48
Bernoulli Naive Bayes	0.36	0.45	0.36	0.34
Multinomial Naive Bayes	0.35	0.28	0.35	0.22

Table 20. Results of 32 × 32 dimension images for RGB vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.46	0.51	0.46	0.43
Logistic	0.43	0.41	0.43	0.37
SVM(linear)	0.43	0.42	0.43	0.35
SVM(poly)	0.59	0.60	0.59	0.59
SVM(RBF)	0.68	0.66	0.68	0.66
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.41	0.46	0.41	0.31
Gaussian Naive Bayes	0.34	0.42	0.34	0.34
Decision Tree	0.40	0.40	0.40	0.40
Bernoulli Naive Bayes	0.31	0.32	0.31	0.16
Multinomial Naive Bayes	0.30	0.19	0.30	0.15

Table 21. Results of 32 × 32 dimension images for Grayscale vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.48	0.48	0.48	0.44
Logistic	0.46	0.44	0.46	0.40
SVM(linear)	0.48	0.44	0.48	0.41
SVM(poly)	0.59	0.59	0.59	0.59
SVM(RBF)	0.67	0.66	0.67	0.65
SVM(sigmoid)	0.30	0.19	0.30	0.14
Random Forest	0.42	0.55	0.42	0.33
Gaussian Naive Bayes	0.35	0.44	0.35	0.34
Decision Tree	0.41	0.40	0.41	0.41
Bernoulli Naive Bayes	0.31	0.26	0.31	0.19
Multinomial Naive Bayes	0.31	0.16	0.31	0.16

Table 22. Results of 32 × 32 dimension images for Red channel vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.45	0.49	0.45	0.42
Logistic	0.41	0.40	0.41	0.34
SVM(linear)	0.41	0.41	0.41	0.32
SVM(poly)	0.59	0.59	0.59	0.59
SVM(RBF)	0.68	0.67	0.68	0.67
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.39	0.34	0.39	0.28
Gaussian Naive Bayes	0.33	0.40	0.33	0.32
Decision Tree	0.39	0.38	0.39	0.38
Bernoulli Naive Bayes	0.32	0.24	0.32	0.18
Multinomial Naive Bayes	0.30	0.19	0.30	0.15

Table 23. Results of 32 × 32 dimension images for Green channel vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.51	0.54	0.51	0.49
Logistic	0.55	0.54	0.55	0.50
SVM(linear)	0.57	0.54	0.57	0.52
SVM(poly)	0.63	0.63	0.63	0.63
SVM(RBF)	0.69	0.67	0.69	0.67
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.50	0.44	0.50	0.40
Gaussian Naive Bayes	0.47	0.53	0.47	0.48
Decision Tree	0.42	0.43	0.42	0.43
Bernoulli Naive Bayes	0.35	0.40	0.35	0.33
Multinomial Naive Bayes	0.33	0.19	0.33	0.19

Table 24. Results of 32 × 32 dimension images for blue channel vector with Normalizer scaling.

7.1.5 PCA of 32×32 dataset

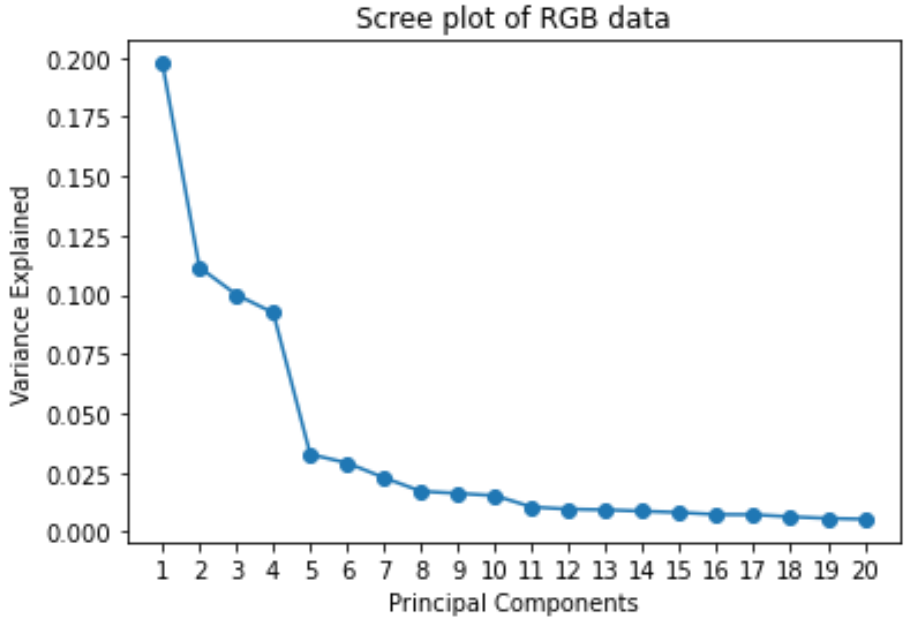


Figure 18. Scree plot of RGB vector of 32×32 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the RGB data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data to models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.62	0.64	0.62	0.59
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.59	0.58	0.59	0.56
SVM(RBF)	0.65	0.64	0.65	0.62
SVM(sigmoid)	0.38	0.40	0.38	0.38
Random Forest	0.56	0.53	0.56	0.51
Gaussian Naive Bayes	0.57	0.55	0.57	0.54
Decision Tree	0.53	0.53	0.53	0.53
Bernoulli Naive Bayes	0.46	0.31	0.46	0.37

Table 25. Results of 32 × 32 dimension images for RGB vector after applying PCA.

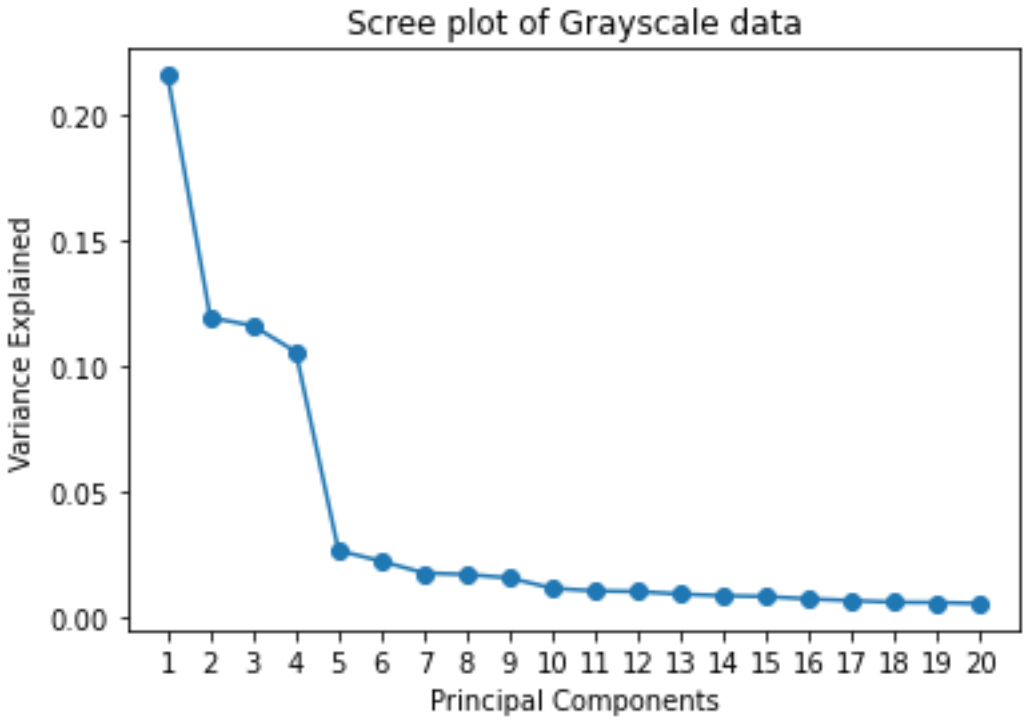


Figure 19. Scree plot of Grayscale vector of 32×32 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the grayscale data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data to models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.49	0.49	0.49	0.47
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.38	0.41	0.38	0.30
SVM(RBF)	0.50	0.50	0.50	0.48
SVM(sigmoid)	0.17	0.15	0.17	0.15
Random Forest	0.42	0.44	0.42	0.38
Gaussian Naive Bayes	0.39	0.40	0.39	0.37
Decision Tree	0.38	0.38	0.38	0.38
Bernoulli Naive Bayes	0.31	0.19	0.31	0.20

Table 26. Results of 32 × 32 dimension images for Grayscale vector after applying PCA.

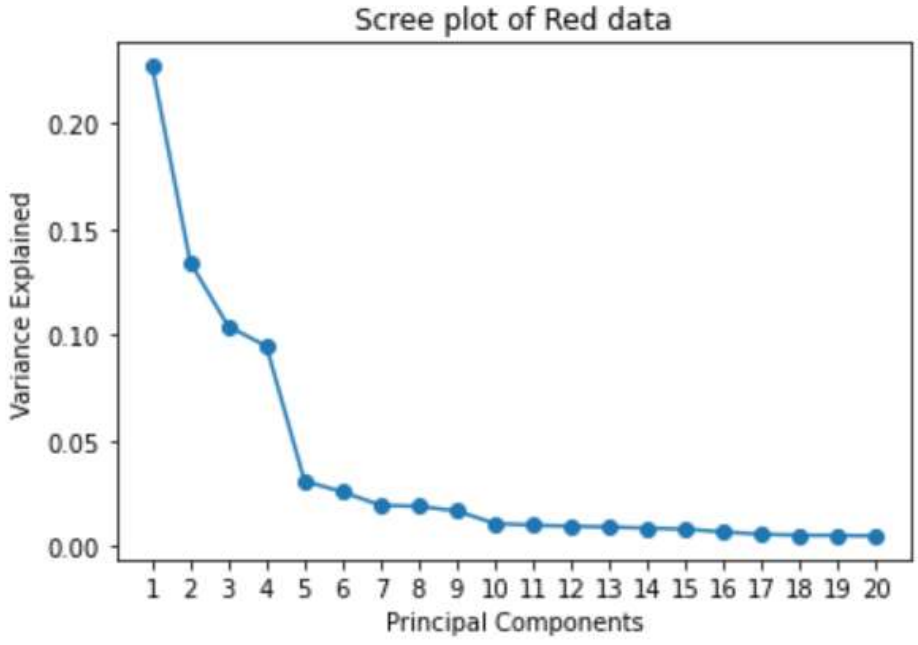


Figure 20. Scree plot of Red channel vector of 32×32 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the Red channel data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data to models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.50	0.49	0.50	0.47
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.39	0.44	0.39	0.33
SVM(RBF)	0.51	0.51	0.51	0.49
SVM(sigmoid)	0.20	0.17	0.20	0.18
Random Forest	0.42	0.41	0.42	0.35
Gaussian Naive Bayes	0.42	0.42	0.42	0.39
Decision Tree	0.39	0.39	0.39	0.39
Bernoulli Naive Bayes	0.32	0.25	0.32	0.26

Table 27. Results of 32 × 32 dimension images for Red channel vector after applying PCA.

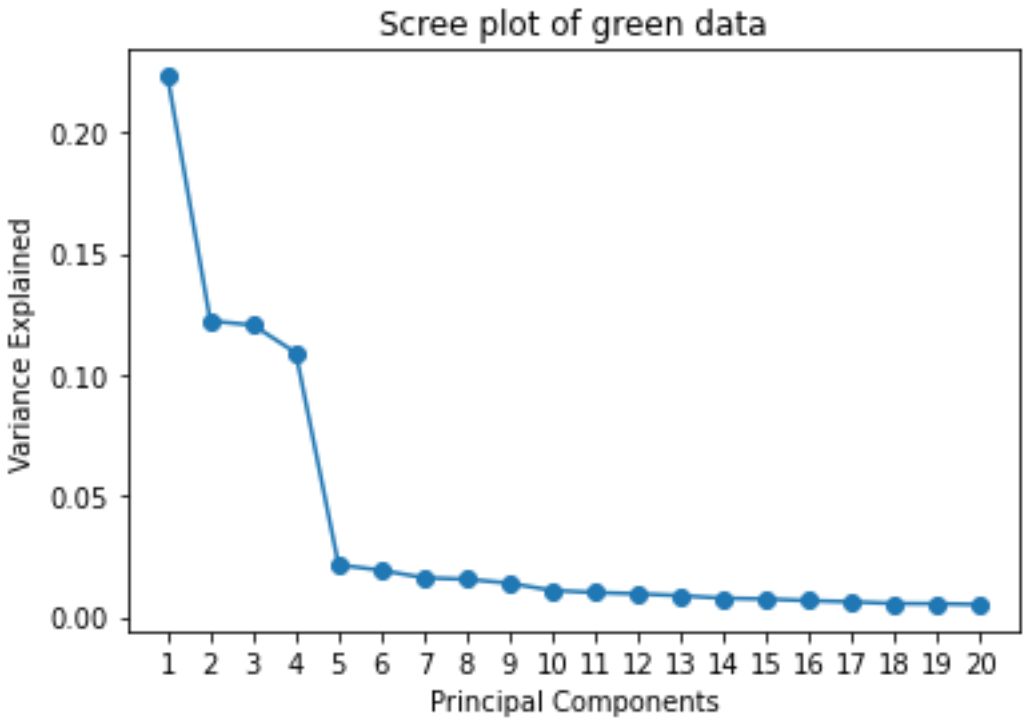


Figure 21. Scree plot of Green channel vector of 32×32 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the Green channel data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data to models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.50	0.51	0.50	0.46
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.35	0.39	0.35	0.25
SVM(RBF)	0.51	0.51	0.51	0.48
SVM(sigmoid)	0.19	0.15	0.19	0.16
Random Forest	0.42	0.51	0.42	0.36
Gaussian Naive Bayes	0.39	0.39	0.39	0.37
Decision Tree	0.40	0.40	0.40	0.40
Bernoulli Naive Bayes	0.30	0.18	0.30	0.19

Table 28. Results of 32 × 32 dimension images for Green channel vector after applying PCA.

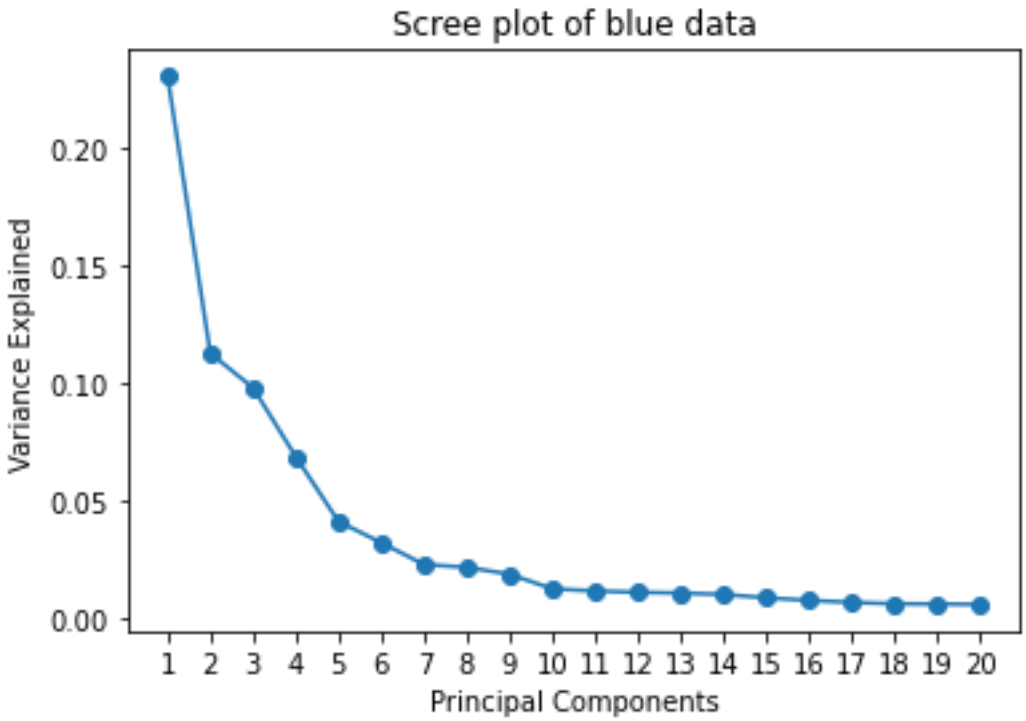


Figure 22. Scree plot of Blue channel vector of 32×32 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the Blue channel data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data to models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.57	0.58	0.57	0.54
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.49	0.46	0.49	0.44
SVM(RBF)	0.58	0.56	0.58	0.56
SVM(sigmoid)	0.29	0.30	0.29	0.29
Random Forest	0.51	0.51	0.51	0.47
Gaussian Naive Bayes	0.50	0.47	0.50	0.47
Decision Tree	0.46	0.47	0.46	0.47
Bernoulli Naive Bayes	0.41	0.26	0.41	0.31

Table 29. Results of 32 × 32 dimension images for Blue channel vector after applying PCA.

7.2 Results of 64 × 64 dimensions images

7.2.1 No Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.54	0.59	0.54	0.53
Logistic	0.68	0.69	0.68	0.68
SVM(linear)	0.76	0.77	0.76	0.76
SVM(poly)	0.81	0.82	0.81	0.81
SVM(RBF)	0.85	0.85	0.85	0.85
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.56	0.56	0.56	0.50
Gaussian Naive Bayes	0.52	0.59	0.52	0.53
Decision Tree	0.49	0.50	0.49	0.50
Bernoulli Naive Bayes	0.38	0.48	0.38	0.36
Multinomial Naive Bayes	0.45	0.54	0.45	0.47

Table 30. Results of 64 × 64 dimension images for RGB vector without scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.43	0.47	0.43	0.41
Logistic	0.68	0.69	0.68	0.68
SVM(linear)	0.76	0.77	0.76	0.76
SVM(poly)	0.57	0.59	0.57	0.57
SVM(RBF)	0.70	0.69	0.70	0.69
SVM(sigmoid)	0.24	0.09	0.24	0.13
Random Forest	0.56	0.56	0.56	0.50
Gaussian Naive Bayes	0.52	0.59	0.52	0.53
Decision Tree	0.50	0.50	0.50	0.50
Bernoulli Naive Bayes	0.31	0.33	0.31	0.16
Multinomial Naive Bayes	0.24	0.32	0.24	0.23

Table 31. Results of 64 × 64 dimension images for Grayscale vector without scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.47	0.49	0.47	0.45
Logistic	0.68	0.69	0.68	0.68
SVM(linear)	0.76	0.77	0.76	0.76
SVM(poly)	0.60	0.61	0.60	0.60
SVM(RBF)	0.71	0.70	0.71	0.69
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.56	0.56	0.56	0.50
Gaussian Naive Bayes	0.52	0.59	0.52	0.53
Decision Tree	0.49	0.50	0.49	0.49
Bernoulli Naive Bayes	0.31	0.28	0.31	0.18
Multinomial Naive Bayes	0.29	0.36	0.29	0.28

Table 32. Results of 64 × 64 dimension images for Red channel vector without scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.41	0.47	0.41	0.39
Logistic	0.68	0.69	0.68	0.68
SVM(linear)	0.76	0.77	0.76	0.76
SVM(poly)	0.57	0.60	0.57	0.58
SVM(RBF)	0.71	0.70	0.71	0.70
SVM(sigmoid)	0.27	0.09	0.27	0.13
Random Forest	0.56	0.56	0.56	0.50
Gaussian Naive Bayes	0.52	0.59	0.52	0.53
Decision Tree	0.49	0.50	0.49	0.49
Bernoulli Naive Bayes	0.31	0.25	0.31	0.17
Multinomial Naive Bayes	0.22	0.33	0.22	0.22

Table 33. Results of 64 × 64 dimension images for Green channel vector without scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.49	0.58	0.49	0.49
Logistic	0.68	0.69	0.68	0.68
SVM(linear)	0.76	0.77	0.76	0.76
SVM(poly)	0.66	0.67	0.66	0.66
SVM(RBF)	0.74	0.74	0.74	0.74
SVM(sigmoid)	0.23	0.12	0.23	0.12
Random Forest	0.56	0.56	0.56	0.50
Gaussian Naive Bayes	0.52	0.59	0.52	0.53
Decision Tree	0.50	0.50	0.50	0.50
Bernoulli Naive Bayes	0.38	0.44	0.38	0.36
Multinomial Naive Bayes	0.44	0.50	0.44	0.45

Table 34. Results of 64 × 64 dimension images for Blue channel vector without scaling.

7.2.2 Standard Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.57	0.63	0.57	0.57
Logistic	0.73	0.74	0.73	0.73
SVM(linear)	0.76	0.77	0.76	0.77
SVM(poly)	0.75	0.76	0.75	0.74
SVM(RBF)	0.86	0.85	0.86	0.85
SVM(sigmoid)	0.58	0.60	0.58	0.57
Random Forest	0.56	0.57	0.56	0.50
Gaussian Naive Bayes	0.52	0.59	0.52	0.53
Decision Tree	0.50	0.50	0.50	0.50
Bernoulli Naive Bayes	0.44	0.52	0.44	0.44

Table 35. Results of 64 × 64 dimension images for RGB vector after standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.47	0.49	0.47	0.45
Logistic	0.38	0.39	0.38	0.38
SVM(linear)	0.38	0.43	0.38	0.39
SVM(poly)	0.55	0.60	0.55	0.53
SVM(RBF)	0.71	0.70	0.71	0.69
SVM(sigmoid)	0.35	0.35	0.35	0.33
Random Forest	0.47	0.52	0.47	0.42
Gaussian Naive Bayes	0.38	0.46	0.38	0.36
Decision Tree	0.42	0.42	0.42	0.42
Bernoulli Naive Bayes	0.31	0.36	0.31	0.27

Table 36. Results of 64 × 64 dimension images for Grayscale vector after standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.50	0.51	0.50	0.48
Logistic	0.43	0.44	0.43	0.43
SVM(linear)	0.43	0.47	0.43	0.44
SVM(poly)	0.59	0.61	0.59	0.57
SVM(RBF)	0.72	0.71	0.72	0.71
SVM(sigmoid)	0.38	0.37	0.38	0.37
Random Forest	0.50	0.53	0.50	0.44
Gaussian Naive Bayes	0.39	0.47	0.39	0.37
Decision Tree	0.45	0.45	0.45	0.45
Bernoulli Naive Bayes	0.34	0.41	0.34	0.31

Table 37. Results of 64 × 64 dimension images for Red channel vector after standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.46	0.49	0.46	0.44
Logistic	0.39	0.41	0.39	0.40
SVM(linear)	0.39	0.44	0.39	0.40
SVM(poly)	0.53	0.58	0.53	0.49
SVM(RBF)	0.71	0.71	0.71	0.70
SVM(sigmoid)	0.33	0.32	0.33	0.31
Random Forest	0.44	0.40	0.44	0.35
Gaussian Naive Bayes	0.37	0.45	0.37	0.35
Decision Tree	0.42	0.42	0.42	0.42
Bernoulli Naive Bayes	0.28	0.30	0.28	0.24

Table 38. Results of 64 × 64 dimension images for Green channel vector after standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.47	0.49	0.47	0.45
Logistic	0.38	0.39	0.38	0.38
SVM(linear)	0.38	0.43	0.38	0.39
SVM(poly)	0.65	0.66	0.65	0.63
SVM(RBF)	0.76	0.75	0.76	0.75
SVM(sigmoid)	0.45	0.45	0.45	0.43
Random Forest	0.47	0.52	0.47	0.42
Gaussian Naive Bayes	0.38	0.46	0.38	0.36
Decision Tree	0.42	0.42	0.42	0.42
Bernoulli Naive Bayes	0.46	0.50	0.46	0.47

Table 39. Results of 64 × 64 dimension images for Blue channel vector after standard scaling.

7.2.3 Minmax Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.55	0.60	0.55	0.54
Logistic	0.75	0.76	0.75	0.75
SVM(linear)	0.75	0.77	0.75	0.76
SVM(poly)	0.80	0.81	0.80	0.81
SVM(RBF)	0.84	0.84	0.84	0.84
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.57	0.57	0.57	0.51
Gaussian Naive Bayes	0.51	0.59	0.51	0.53
Decision Tree	0.48	0.50	0.48	0.49
Bernoulli Naive Bayes	0.38	0.49	0.38	0.36
Multinomial Naive Bayes	0.38	0.49	0.38	0.36

Table 40. Results of 64 × 64 dimension images for RGB vector after Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.43	0.46	0.43	0.41
Logistic	0.44	0.45	0.44	0.43
SVM(linear)	0.45	0.48	0.45	0.45
SVM(poly)	0.53	0.58	0.53	0.54
SVM(RBF)	0.69	0.68	0.69	0.68
SVM(sigmoid)	0.23	0.09	0.23	0.13
Random Forest	0.46	0.50	0.46	0.40
Gaussian Naive Bayes	0.37	0.46	0.37	0.35
Decision Tree	0.37	0.38	0.37	0.38
Bernoulli Naive Bayes	0.31	0.30	0.31	0.16
Multinomial Naive Bayes	0.31	0.30	0.31	0.16

Table 41. Results of 64 × 64 dimension images for grayscale vector after Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.48	0.49	0.48	0.46
Logistic	0.51	0.50	0.51	0.50
SVM(linear)	0.48	0.50	0.48	0.49
SVM(poly)	0.56	0.60	0.56	0.57
SVM(RBF)	0.70	0.69	0.70	0.69
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.50	0.53	0.50	0.45
Gaussian Naive Bayes	0.38	0.48	0.38	0.36
Decision Tree	0.41	0.43	0.41	0.42
Bernoulli Naive Bayes	0.31	0.28	0.31	0.18
Multinomial Naive Bayes	0.31	0.28	0.31	0.18

Table 42. Results of 64 × 64 dimension images for Red channel vector after Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.43	0.46	0.43	0.41
Logistic	0.47	0.48	0.47	0.47
SVM(linear)	0.45	0.49	0.45	0.45
SVM(poly)	0.57	0.59	0.57	0.57
SVM(RBF)	0.70	0.69	0.70	0.69
SVM(sigmoid)	0.27	0.09	0.27	0.13
Random Forest	0.43	0.42	0.43	0.35
Gaussian Naive Bayes	0.35	0.43	0.35	0.34
Decision Tree	0.40	0.41	0.40	0.40
Bernoulli Naive Bayes	0.31	0.26	0.31	0.18
Multinomial Naive Bayes	0.31	0.26	0.31	0.18

Table 43. Results of 64 × 64 dimension images for Green channel vector after Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.48	0.58	0.48	0.48
Logistic	0.56	0.57	0.56	0.56
SVM(linear)	0.52	0.56	0.52	0.53
SVM(poly)	0.64	0.65	0.64	0.64
SVM(RBF)	0.74	0.73	0.74	0.73
SVM(sigmoid)	0.25	0.11	0.25	0.13
Random Forest	0.56	0.52	0.56	0.50
Gaussian Naive Bayes	0.52	0.58	0.52	0.53
Decision Tree	0.42	0.45	0.42	0.43
Bernoulli Naive Bayes	0.38	0.44	0.38	0.36
Multinomial Naive Bayes	0.38	0.44	0.38	0.36

Table 44. Results of 64 × 64 dimension images for Blue channel vector after Minmax scaling.

7.2.4 Normalizer Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.52	0.58	0.52	0.51
Logistic	0.64	0.62	0.64	0.61
SVM(linear)	0.69	0.66	0.69	0.66
SVM(poly)	0.83	0.83	0.83	0.83
SVM(RBF)	0.84	0.83	0.84	0.83
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.53	0.47	0.53	0.45
Gaussian Naive Bayes	0.49	0.58	0.49	0.51
Decision Tree	0.48	0.49	0.48	0.49
Bernoulli Naive Bayes	0.38	0.48	0.38	0.36
Multinomial Naive Bayes	0.46	0.40	0.46	0.37

Table 45. Results of 64 × 64 dimension images for RGB vector after Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.40	0.47	0.40	0.37
Logistic	0.44	0.42	0.44	0.37
SVM(linear)	0.44	0.43	0.44	0.36
SVM(poly)	0.58	0.60	0.58	0.59
SVM(RBF)	0.68	0.66	0.68	0.67
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.42	0.46	0.42	0.32
Gaussian Naive Bayes	0.35	0.44	0.35	0.35
Decision Tree	0.38	0.39	0.38	0.38
Bernoulli Naive Bayes	0.31	0.33	0.31	0.16
Multinomial Naive Bayes	0.31	0.16	0.31	0.16

Table 46. Results of 64 × 64 dimension images for Grayscale vector after Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.43	0.48	0.43	0.41
Logistic	0.47	0.47	0.47	0.42
SVM(linear)	0.49	0.46	0.49	0.43
SVM(poly)	0.61	0.62	0.61	0.61
SVM(RBF)	0.69	0.68	0.69	0.67
SVM(sigmoid)	0.30	0.19	0.30	0.14
Random Forest	0.43	0.46	0.43	0.34
Gaussian Naive Bayes	0.37	0.46	0.37	0.36
Decision Tree	0.41	0.41	0.41	0.41
Bernoulli Naive Bayes	0.31	0.28	0.31	0.18
Multinomial Naive Bayes	0.33	0.20	0.33	0.21

Table 47. Results of 64 × 64 dimension images for Red channel vector after Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.39	0.45	0.39	0.36
Logistic	0.42	0.40	0.42	0.35
SVM(linear)	0.41	0.40	0.41	0.32
SVM(poly)	0.59	0.61	0.59	0.60
SVM(RBF)	0.69	068.	0.69	0.68
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.40	0.34	0.40	0.29
Gaussian Naive Bayes	0.34	0.42	0.34	0.33
Decision Tree	0.40	0.40	0.40	0.40
Bernoulli Naive Bayes	0.31	0.25	0.31	0.17
Multinomial Naive Bayes	0.30	0.19	0.30	0.15

Table 48. Results of 64 × 64 dimension images for Green channel vector after Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.44	0.54	0.44	0.44
Logistic	0.56	0.54	0.56	0.51
SVM(linear)	0.58	0.56	0.58	0.54
SVM(poly)	0.67	0.68	0.67	0.67
SVM(RBF)	0.71	0.70	0.71	0.70
SVM(sigmoid)	0.29	0.09	0.29	0.13
Random Forest	0.49	0.38	0.49	0.40
Gaussian Naive Bayes	0.49	0.55	0.49	0.50
Decision Tree	0.42	0.43	0.42	0.43
Bernoulli Naive Bayes	0.38	0.44	0.38	0.36
Multinomial Naive Bayes	0.40	0.35	0.40	0.29

Table 49. Results of 64 × 64 dimension images for Blue channel vector after Normalizer scaling.

7.2.5 PCA on 64×64 data

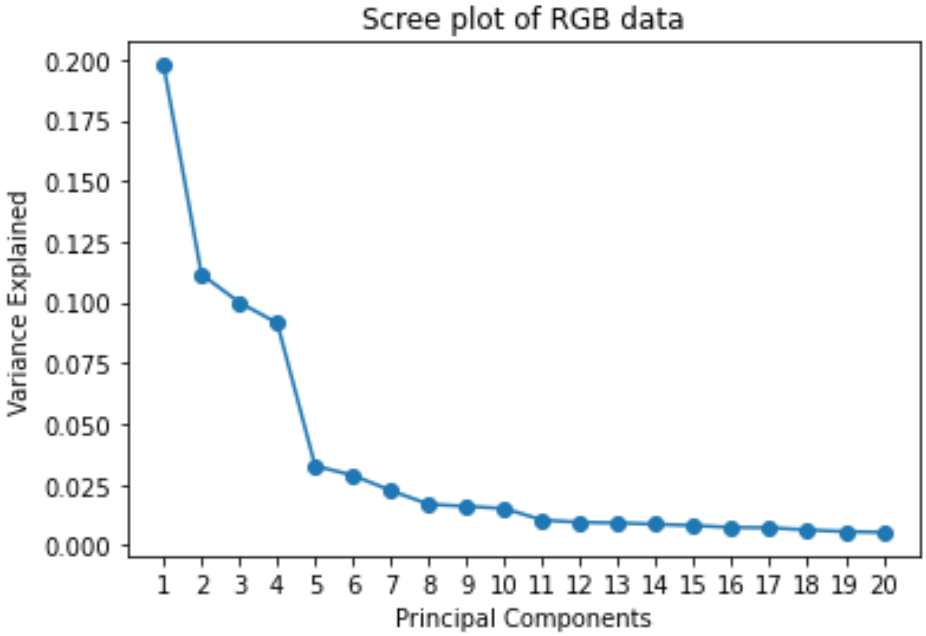


Figure 23. Scree plot of RGB vector of 64×64 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the RGB data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data with models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.62	0.64	0.62	0.59
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.59	0.58	0.59	0.56
SVM(RBF)	0.65	0.65	0.65	0.63
SVM(sigmoid)	0.37	0.39	0.37	0.37
Random Forest	0.56	0.53	0.56	0.51
Gaussian Naive Bayes	0.57	0.55	0.57	0.54
Decision Tree	0.53	0.53	0.53	0.53
Bernoulli Naive Bayes	0.43	0.38	0.43	0.39

Table 50. Results of 64 × 64 dimension images for RGB vector after applying PCA.

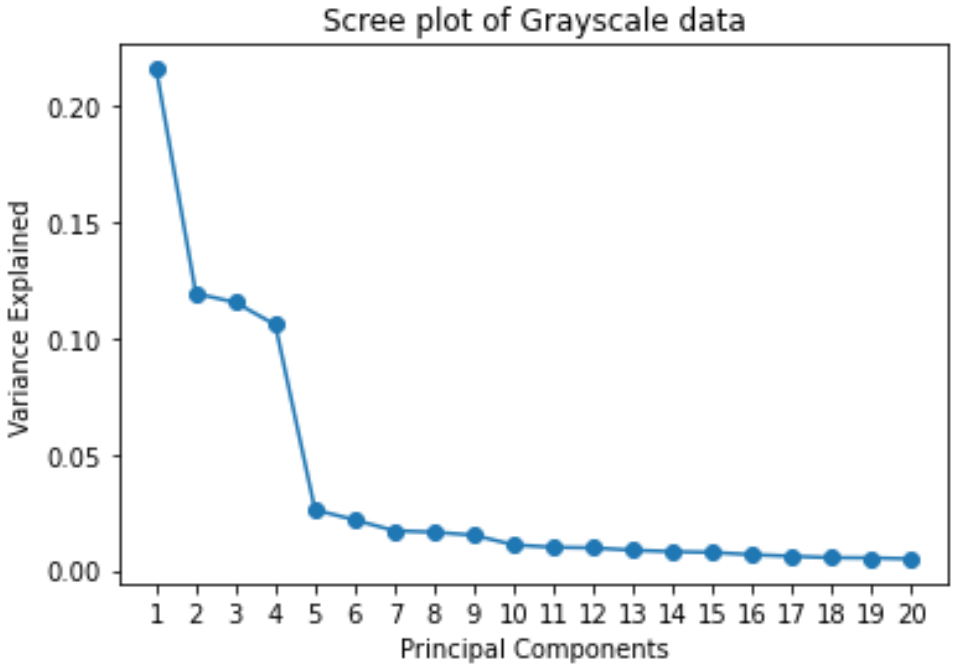


Figure 24. Scree plot of Grayscale vector of 64×64 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the grayscale data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data with models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.62	0.64	0.62	0.59
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.38	0.42	0.38	0.30
SVM(RBF)	0.50	0.50	0.50	0.48
SVM(sigmoid)	0.18	0.15	0.18	0.16
Random Forest	0.56	0.53	0.56	0.51
Gaussian Naive Bayes	0.57	0.55	0.57	0.54
Decision Tree	0.52	0.52	0.52	0.52
Bernoulli Naive Bayes	0.31	0.19	0.31	0.21

Table 51. Results of 64 × 64 dimension images for Grayscale vector after applying PCA.

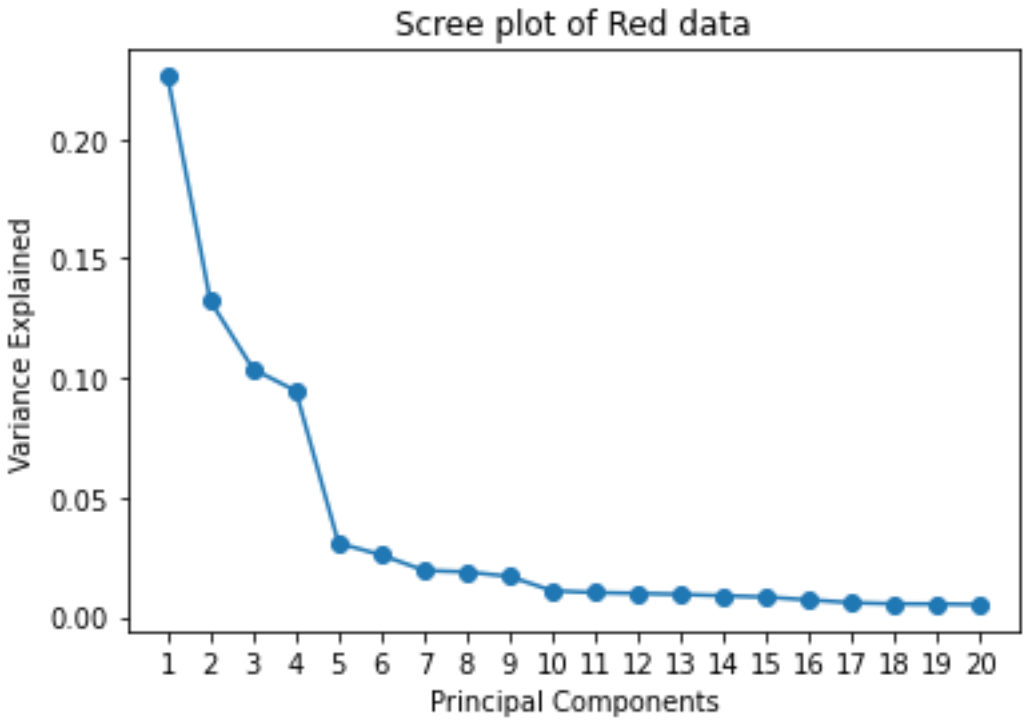


Figure 25. Scree plot of Red channel vector of 64×64 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the Red channel data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data with models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.62	0.64	0.62	0.59
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.39	0.43	0.39	0.33
SVM(RBF)	0.52	0.52	0.52	0.50
SVM(sigmoid)	0.20	0.17	0.20	0.18
Random Forest	0.56	0.53	0.56	0.51
Gaussian Naive Bayes	0.57	0.55	0.57	0.54
Decision Tree	0.52	0.52	0.52	0.52
Bernoulli Naive Bayes	0.33	0.25	0.33	0.26

Table 52. Results of 64 × 64 dimension images for Red channel vector after applying PCA.

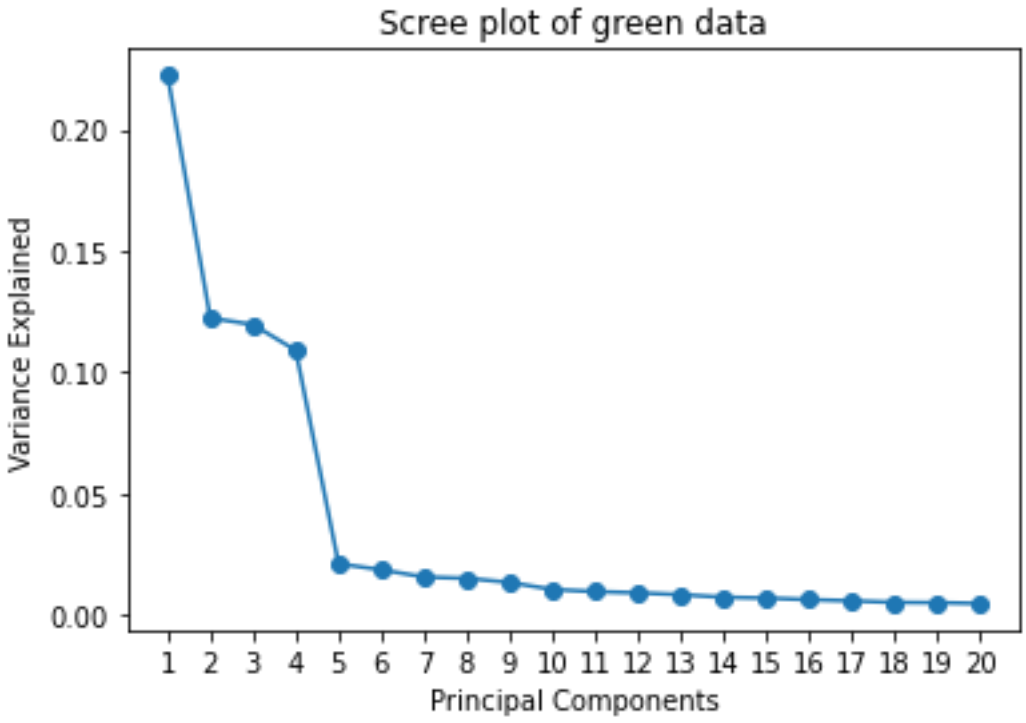


Figure 26. Scree plot of Green channel vector of 64×64 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the green channel data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data with models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.62	0.64	0.62	0.59
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.35	0.38	0.35	0.26
SVM(RBF)	0.51	0.51	0.51	0.48
SVM(sigmoid)	0.18	0.15	0.18	0.16
Random Forest	0.56	0.53	0.56	0.51
Gaussian Naive Bayes	0.57	0.55	0.57	0.54
Decision Tree	0.53	0.52	0.53	0.52
Bernoulli Naive Bayes	0.31	0.18	0.31	0.19

Table 53. Results of 64 × 64 dimension images for Green channel vector after applying PCA.

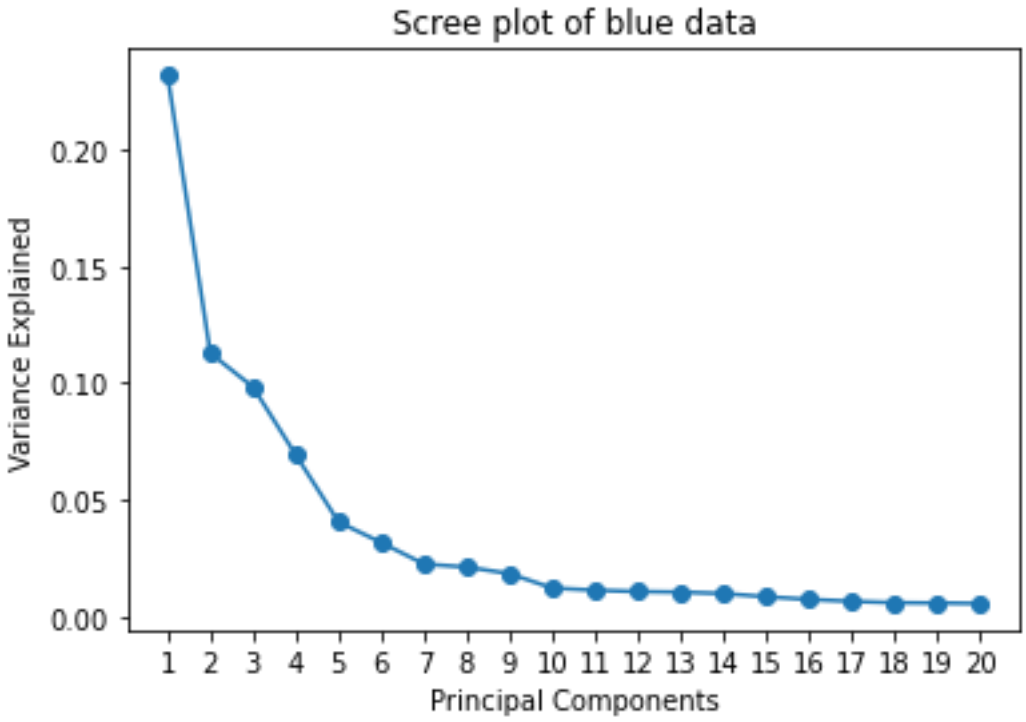


Figure 27. Scree plot of Blue channel vector of 64×64 images.

The elbow method shows that 5 components are better for analysis and the same is applied to the blue channel data. The process is Standard scaling then PCA and ICA. Lastly, fitting the data with models.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.62	0.64	0.62	0.59
Logistic	0.29	0.09	0.29	0.13
SVM(linear)	0.29	0.09	0.29	0.13
SVM(poly)	0.49	0.46	0.49	0.44
SVM(RBF)	0.58	0.56	0.58	0.55
SVM(sigmoid)	0.28	0.28	0.28	0.28
Random Forest	0.56	0.53	0.56	0.51
Gaussian Naive Bayes	0.57	0.55	0.57	0.54
Decision Tree	0.52	0.52	0.52	0.52
Bernoulli Naive Bayes	0.41	0.26	0.41	0.31

Table 54. Results of 64 × 64 dimension images for Blue channel vector after applying PCA.

7.3 Results of balanced dataset(32 × 32)

7.3.1 No scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.45	0.59	0.45	0.42
Logistic	0.56	0.56	0.56	0.56
SVM(linear)	0.69	0.69	0.69	0.69
SVM(poly)	0.72	0.72	0.72	0.72
SVM(RBF)	0.76	0.75	0.76	0.75
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.54	0.55	0.54	0.50
Gaussian Naive Bayes	0.46	0.48	0.46	0.45
Decision Tree	0.44	0.44	0.44	0.44
Bernoulli Naive Bayes	0.24	0.32	0.24	0.20
Multinomial Naive Bayes	0.41	0.43	0.41	0.40

Table 55. Results of 32 × 32 dimension images for RGB vector with no scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.49	0.50	0.49	0.49
Logistic	0.56	0.56	0.56	0.56
SVM(linear)	0.69	0.69	0.69	0.69
SVM(poly)	0.49	0.50	0.49	0.49
SVM(RBF)	0.59	0.59	0.59	0.59
SVM(sigmoid)	0.08	0.02	0.08	0.03
Random Forest	0.54	0.55	0.54	0.50
Gaussian Naive Bayes	0.46	0.48	0.46	0.45
Decision Tree	0.44	0.44	0.44	0.44
Bernoulli Naive Bayes	0.10	0.20	0.10	0.03
Multinomial Naive Bayes	0.30	0.31	0.30	0.28

Table 56. Results of 32 × 32 dimension images for Grayscale vector with no scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.45	0.59	0.45	0.42
Logistic	0.56	0.56	0.56	0.56
SVM(linear)	0.69	0.69	0.69	0.69
SVM(poly)	0.52	0.52	0.52	0.52
SVM(RBF)	0.61	0.61	0.61	0.60
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.54	0.55	0.54	0.50
Gaussian Naive Bayes	0.46	0.48	0.46	0.45
Decision Tree	0.43	0.44	0.43	0.44
Bernoulli Naive Bayes	0.15	0.26	0.15	0.10
Multinomial Naive Bayes	0.30	0.30	0.30	0.28

Table 57. Results of 32 × 32 dimension images for Red vector with no scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.45	0.59	0.45	0.42
Logistic	0.56	0.56	0.56	0.56
SVM(linear)	0.69	0.69	0.69	0.69
SVM(poly)	0.48	0.49	0.48	0.48
SVM(RBF)	0.61	0.61	0.61	0.60
SVM(sigmoid)	0.08	0.02	0.08	0.03
Random Forest	0.54	0.55	0.54	0.50
Gaussian Naive Bayes	0.46	0.48	0.46	0.45
Decision Tree	0.44	0.44	0.44	0.44
Bernoulli Naive Bayes	0.14	0.24	0.14	0.08
Multinomial Naive Bayes	0.28	0.29	0.28	0.27

Table 58. Results of 32 × 32 dimension images for Green vector with no scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.45	0.59	0.45	0.42
Logistic	0.56	0.56	0.56	0.56
SVM(linear)	0.69	0.69	0.69	0.69
SVM(poly)	0.54	0.55	0.54	0.54
SVM(RBF)	0.64	0.64	0.64	0.64
SVM(sigmoid)	0.09	0.04	0.09	0.03
Random Forest	0.54	0.55	0.54	0.50
Gaussian Naive Bayes	0.46	0.48	0.46	0.45
Decision Tree	0.44	0.44	0.44	0.44
Bernoulli Naive Bayes	0.23	0.27	0.23	0.20
Multinomial Naive Bayes	0.39	0.38	0.39	0.36

Table 59. Results of 32 × 32 dimension images for blue vector with no scaling.

7.3.2 Standard Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.43	0.59	0.43	0.40
Logistic	0.63	0.63	0.63	0.63
SVM(linear)	0.70	0.70	0.70	0.70
SVM(poly)	0.69	0.72	0.69	0.68
SVM(RBF)	0.77	0.77	0.77	0.77
SVM(sigmoid)	0.47	0.53	0.47	0.48
Random Forest	0.55	0.57	0.55	0.51
Gaussian Naive Bayes	0.46	0.47	0.46	0.45
Decision Tree	0.44	0.44	0.44	0.44
Bernoulli Naive Bayes	0.42	0.43	0.42	0.41

Table 60. Results of 32 × 32 dimension images for RGB vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.37	0.37	0.37	0.34
Logistic	0.31	0.31	0.31	0.31
SVM(linear)	0.36	0.36	0.36	0.36
SVM(poly)	0.51	0.53	0.51	0.48
SVM(RBF)	0.63	0.64	0.63	0.63
SVM(sigmoid)	0.29	0.33	0.29	0.29
Random Forest	0.47	0.44	0.47	0.44
Gaussian Naive Bayes	0.41	0.41	0.41	0.39
Decision Tree	0.34	0.35	0.34	0.34
Bernoulli Naive Bayes	0.34	0.34	0.34	0.32

Table 61. Results of 32 × 32 dimension images for Grayscale vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.41	0.38	0.41	0.37
Logistic	0.33	0.33	0.33	0.33
SVM(linear)	0.38	0.39	0.38	0.38
SVM(poly)	0.54	0.57	0.54	0.52
SVM(RBF)	0.65	0.64	0.65	0.64
SVM(sigmoid)	0.31	0.34	0.31	0.31
Random Forest	0.47	0.44	0.47	0.43
Gaussian Naive Bayes	0.44	0.43	0.44	0.42
Decision Tree	0.36	0.36	0.36	0.36
Bernoulli Naive Bayes	0.36	0.35	0.36	0.33

Table 62. Results of 32 × 32 dimension images for Red channel vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.34	0.35	0.34	0.30
Logistic	0.33	0.33	0.33	0.33
SVM(linear)	0.37	0.38	0.37	0.37
SVM(poly)	0.48	0.52	0.48	0.45
SVM(RBF)	0.63	0.63	0.63	0.63
SVM(sigmoid)	0.29	0.32	0.29	0.29
Random Forest	0.45	0.44	0.45	0.42
Gaussian Naive Bayes	0.39	0.39	0.39	0.37
Decision Tree	0.33	0.33	0.33	0.33
Bernoulli Naive Bayes	0.32	0.32	0.32	0.30

Table 63. Results of 32 × 32 dimension images for Green channel vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.37	0.37	0.34	0.34
Logistic	0.31	0.31	0.31	0.31
SVM(linear)	0.36	0.36	0.36	0.36
SVM(poly)	0.51	0.53	0.51	0.48
SVM(RBF)	0.63	0.64	0.63	0.63
SVM(sigmoid)	0.29	0.33	0.29	0.29
Random Forest	0.47	0.45	0.47	0.44
Gaussian Naive Bayes	0.41	0.41	0.41	0.39
Decision Tree	0.35	0.35	0.35	0.35
Bernoulli Naive Bayes	0.34	0.34	0.34	0.32

Table 64. Results of 32 × 32 dimension images for Blue channel vector with Standard scaling.

7.3.3 Minmax Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.48	0.60	0.48	0.45
Logistic	0.65	0.65	0.65	0.64
SVM(linear)	0.67	0.68	0.67	0.67
SVM(poly)	0.71	0.72	0.71	0.71
SVM(RBF)	0.74	0.75	0.74	0.74
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.50	0.52	0.50	0.47
Gaussian Naive Bayes	0.39	0.46	0.39	0.40
Decision Tree	0.37	0.39	0.37	0.37
Bernoulli Naive Bayes	0.24	0.31	0.24	0.20
Multinomial Naive Bayes	0.43	0.43	0.43	0.41

Table 65. Results of 32 × 32 dimension images for RGB vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.41	0.45	0.41	0.37
Logistic	0.40	0.39	0.40	0.39
SVM(linear)	0.41	0.42	0.41	0.41
SVM(poly)	0.46	0.48	0.46	0.46
SVM(RBF)	0.57	0.58	0.57	0.56
SVM(sigmoid)	0.08	0.02	0.08	0.03
Random Forest	0.45	0.44	0.45	0.41
Gaussian Naive Bayes	0.37	0.40	0.37	0.36
Decision Tree	0.31	0.32	0.31	0.31
Bernoulli Naive Bayes	0.12	0.19	0.12	0.05
Multinomial Naive Bayes	0.30	0.30	0.30	0.28

Table 66. Results of 32 × 32 dimension images for Grayscale channel vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.45	0.43	0.45	0.41
Logistic	0.41	0.41	0.41	0.41
SVM(linear)	0.42	0.43	0.42	0.42
SVM(poly)	0.50	0.52	0.50	0.49
SVM(RBF)	0.58	0.60	0.58	0.58
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.47	0.45	0.47	0.43
Gaussian Naive Bayes	0.40	0.43	0.40	0.39
Decision Tree	0.32	0.34	0.32	0.32
Bernoulli Naive Bayes	0.15	0.26	0.15	0.10
Multinomial Naive Bayes	0.31	0.31	0.31	0.29

Table 67. Results of 32 × 32 dimension images for Red channel vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.38	0.40	0.38	0.34
Logistic	0.40	0.39	0.40	0.38
SVM(linear)	0.42	0.42	0.42	0.41
SVM(poly)	0.46	0.47	0.46	0.45
SVM(RBF)	0.56	0.57	0.56	0.56
SVM(sigmoid)	0.08	0.02	0.08	0.03
Random Forest	0.42	0.43	0.42	0.40
Gaussian Naive Bayes	0.36	0.38	0.36	0.35
Decision Tree	0.29	0.30	0.29	0.29
Bernoulli Naive Bayes	0.14	0.28	0.14	0.08
Multinomial Naive Bayes	0.29	0.30	0.29	0.27

Table 68. Results of 32 × 32 dimension images for Green channel vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.48	0.50	0.48	0.45
Logistic	0.47	0.48	0.47	0.47
SVM(linear)	0.45	0.46	0.45	0.45
SVM(poly)	0.50	0.53	0.50	0.53
SVM(RBF)	0.63	0.64	0.63	0.63
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.51	0.51	0.51	0.48
Gaussian Naive Bayes	0.46	0.50	0.46	0.46
Decision Tree	0.33	0.34	0.33	0.33
Bernoulli Naive Bayes	0.23	0.27	0.23	0.20
Multinomial Naive Bayes	0.39	0.37	0.39	0.37

Table 69. Results of 32 × 32 dimension images for Blue channel vector with Minmax scaling.

7.3.4 Normalizer Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.44	0.55	0.44	0.41
Logistic	0.52	0.52	0.52	0.50
SVM(linear)	0.57	0.57	0.57	0.56
SVM(poly)	0.72	0.72	0.72	0.72
SVM(RBF)	0.73	0.73	0.73	0.73
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.53	0.54	0.53	0.50
Gaussian Naive Bayes	0.46	0.48	0.46	0.46
Decision Tree	0.41	0.41	0.41	0.40
Bernoulli Naive Bayes	0.24	0.32	0.24	0.20
Multinomial Naive Bayes	0.42	0.40	0.42	0.40

Table 70. Results of 32 × 32 dimension images for RGB vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.35	0.35	0.35	0.31
Logistic	0.37	0.35	0.37	0.34
SVM(linear)	0.37	0.36	0.37	0.34
SVM(poly)	0.49	0.49	0.49	0.49
SVM(RBF)	0.58	0.58	0.58	0.57
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.44	0.41	0.44	0.41
Gaussian Naive Bayes	0.39	0.39	0.39	0.37
Decision Tree	0.32	0.33	0.32	0.33
Bernoulli Naive Bayes	0.10	0.20	0.10	0.03
Multinomial Naive Bayes	0.29	0.30	0.29	0.28

Table 71. Results of 32 × 32 dimension images for Grayscale vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.38	0.37	0.37	0.38
Logistic	0.39	0.35	0.39	0.36
SVM(linear)	0.40	0.38	0.40	0.37
SVM(poly)	0.50	0.51	0.50	0.50
SVM(RBF)	0.58	0.58	0.58	0.58
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.46	0.47	0.46	0.43
Gaussian Naive Bayes	0.39	0.39	0.39	0.37
Decision Tree	0.33	0.34	0.33	0.33
Bernoulli Naive Bayes	0.15	0.26	0.15	0.10
Multinomial Naive Bayes	0.32	0.32	0.32	0.31

Table 72. Results of 32 × 32 dimension images for Red channel vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.33	0.35	0.33	0.29
Logistic	0.37	0.35	0.37	0.34
SVM(linear)	0.37	0.35	0.37	0.34
SVM(poly)	0.50	0.50	0.50	0.50
SVM(RBF)	0.57	0.58	0.57	0.57
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.46	0.45	0.46	0.45
Gaussian Naive Bayes	0.37	0.37	0.37	0.35
Decision Tree	0.30	0.30	0.30	0.30
Bernoulli Naive Bayes	0.14	0.24	0.14	0.08
Multinomial Naive Bayes	0.27	0.29	0.27	0.27

Table 73. Results of 32 × 32 dimension images for Green channel vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.41	0.50	0.41	0.38
Logistic	0.47	0.46	0.47	0.45
SVM(linear)	0.49	0.49	0.49	0.47
SVM(poly)	0.53	0.54	0.53	0.53
SVM(RBF)	0.58	0.58	0.58	0.58
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.48	0.45	0.48	0.44
Gaussian Naive Bayes	0.46	0.46	0.46	0.45
Decision Tree	0.35	0.36	0.35	0.35
Bernoulli Naive Bayes	0.23	0.27	0.23	0.20
Multinomial Naive Bayes	0.39	0.37	0.39	0.37

Table 74. Results of 32 × 32 dimension images for Blue channel vector with Normalizer scaling.

7.4 Results of balanced data with 64 × 64 dimension

7.4.1 No Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.42	0.58	0.42	0.39
Logistic	0.62	0.62	0.62	0.62
SVM(linear)	0.70	0.70	0.70	0.70
SVM(poly)	0.72	0.73	0.72	0.72
SVM(RBF)	0.76	0.75	0.76	0.75
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.54	0.56	0.54	0.50
Gaussian Naive Bayes	0.45	0.47	0.45	0.45
Decision Tree	0.39	0.40	0.39	0.40
Bernoulli Naive Bayes	0.28	0.41	0.28	0.24
Multinomial Naive Bayes	0.42	0.44	0.42	0.41

Table 75. Results of 64 × 64-dimension images for RGB vector with No scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.42	0.58	0.42	0.39
Logistic	0.62	0.62	0.62	0.62
SVM(linear)	0.70	0.70	0.70	0.70
SVM(poly)	0.49	0.49	0.49	0.49
SVM(RBF)	0.59	0.58	0.59	0.58
SVM(sigmoid)	0.08	0.02	0.08	0.03
Random Forest	0.54	0.56	0.54	0.50
Gaussian Naive Bayes	0.45	0.47	0.45	0.45
Decision Tree	0.40	0.40	0.40	0.40
Bernoulli Naive Bayes	0.11	0.15	0.11	0.03
Multinomial Naive Bayes	0.30	0.30	0.30	0.28

Table 76. Results of 64 × 64 dimension images for Grayscale vector with No scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.42	0.58	0.42	0.39
Logistic	0.62	0.62	0.62	0.62
SVM(linear)	0.70	0.70	0.70	0.70
SVM(poly)	0.50	0.51	0.50	0.50
SVM(RBF)	0.60	0.60	0.60	0.60
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.54	0.56	0.54	0.50
Gaussian Naive Bayes	0.45	0.47	0.45	0.45
Decision Tree	0.40	0.40	0.40	0.40
Bernoulli Naive Bayes	0.13	0.27	0.13	0.08
Multinomial Naive Bayes	0.31	0.31	0.31	0.29

Table 77. Results of 64 × 64 dimension images for Red channel vector with No scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.42	0.58	0.42	0.39
Logistic	0.62	0.62	0.62	0.62
SVM(linear)	0.70	0.70	0.70	0.70
SVM(poly)	0.47	0.47	0.47	0.47
SVM(RBF)	0.59	0.59	0.59	0.59
SVM(sigmoid)	0.08	0.02	0.08	0.03
Random Forest	0.54	0.56	0.54	0.50
Gaussian Naive Bayes	0.45	0.47	0.45	0.45
Decision Tree	0.40	0.40	0.40	0.40
Bernoulli Naive Bayes	0.13	0.32	0.13	0.07
Multinomial Naive Bayes	0.27	0.28	0.27	0.25

Table 78. Results of 64 × 64 dimension images for Green channel vector with No scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.42	0.58	0.42	0.39
Logistic	0.62	0.62	0.62	0.62
SVM(linear)	0.70	0.70	0.70	0.70
SVM(poly)	0.55	0.56	0.55	0.55
SVM(RBF)	0.64	0.64	0.64	0.64
SVM(sigmoid)	0.09	0.04	0.09	0.03
Random Forest	0.54	0.56	0.54	0.50
Gaussian Naive Bayes	0.45	0.47	0.45	0.45
Decision Tree	0.39	0.39	0.39	0.38
Bernoulli Naive Bayes	0.28	0.34	0.28	0.25
Multinomial Naive Bayes	0.40	0.39	0.40	0.37

Table 79. Results of 64 × 64 dimension images for Blue channel vector with No scaling.

7.4.2 Standard Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.43	0.59	0.43	0.41
Logistic	0.66	0.66	0.66	0.66
SVM(linear)	0.72	0.72	0.72	0.72
SVM(poly)	0.67	0.71	0.67	0.66
SVM(RBF)	0.77	0.77	0.77	0.77
SVM(sigmoid)	0.50	0.55	0.50	0.51
Random Forest	0.54	0.57	0.54	0.51
Gaussian Naive Bayes	0.46	0.47	0.46	0.45
Decision Tree	0.39	0.40	0.39	0.39
Bernoulli Naive Bayes	0.41	0.43	0.41	0.40

Table 80. Results of 64 × 64 dimension images for RGB vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.36	0.34	0.36	0.32
Logistic	0.32	0.32	0.32	0.31
SVM(linear)	0.39	0.39	0.39	0.39
SVM(poly)	0.50	0.53	0.50	0.47
SVM(RBF)	0.63	0.63	0.63	0.63
SVM(sigmoid)	0.28	0.33	0.28	0.28
Random Forest	0.47	0.45	0.47	0.44
Gaussian Naive Bayes	0.42	0.42	0.42	0.40
Decision Tree	0.34	0.34	0.34	0.34
Bernoulli Naive Bayes	0.34	0.34	0.34	0.32

Table 81. Results of 64 × 64 dimension images for Grayscale vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.41	0.38	0.41	0.37
Logistic	0.33	0.34	0.33	0.33
SVM(linear)	0.39	0.40	0.39	0.39
SVM(poly)	0.53	0.57	0.53	0.51
SVM(RBF)	0.64	0.64	0.64	0.64
SVM(sigmoid)	0.30	0.36	0.30	0.30
Random Forest	0.49	0.46	0.49	0.45
Gaussian Naive Bayes	0.43	0.43	0.43	0.42
Decision Tree	0.36	0.37	0.36	0.36
Bernoulli Naive Bayes	0.35	0.34	0.35	0.32

Table 82. Results of 64 × 64 dimension images for Red channel vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.33	0.34	0.33	0.29
Logistic	0.31	0.31	0.31	0.31
SVM(linear)	0.38	0.38	0.38	0.38
SVM(poly)	0.47	0.51	0.47	0.44
SVM(RBF)	0.62	0.63	0.62	0.62
SVM(sigmoid)	0.25	0.30	0.25	0.25
Random Forest	0.43	0.42	0.43	0.41
Gaussian Naive Bayes	0.39	0.38	0.39	0.37
Decision Tree	0.32	0.32	0.32	0.32
Bernoulli Naive Bayes	0.32	0.33	0.32	0.30

Table 83. Results of 64 × 64 dimension images for Green channel vector with Standard scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.36	0.34	0.36	0.32
Logistic	0.32	0.32	0.32	0.31
SVM(linear)	0.39	0.39	0.39	0.39
SVM(poly)	0.50	0.53	0.50	0.47
SVM(RBF)	0.63	0.63	0.63	0.63
SVM(sigmoid)	0.28	0.33	0.28	0.28
Random Forest	0.47	0.45	0.47	0.44
Gaussian Naive Bayes	0.42	0.42	0.42	0.40
Decision Tree	0.35	0.35	0.35	0.35
Bernoulli Naive Bayes	0.34	0.34	0.34	0.32

Table 84. Results of 64 × 64 dimension images for Blue channel vector with Standard scaling.

7.4.3 Minmax Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.45	0.48	0.45	0.42
Logistic	0.65	0.66	0.65	0.65
SVM(linear)	0.68	0.69	0.68	0.68
SVM(poly)	0.71	0.72	0.71	0.71
SVM(RBF)	0.73	0.74	0.73	0.73
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.49	0.52	0.49	0.47
Gaussian Naive Bayes	0.40	0.46	0.40	0.41
Decision Tree	0.35	0.36	0.35	0.35
Bernoulli Naive Bayes	0.28	0.44	0.28	0.25
Multinomial Naive Bayes	0.43	0.44	0.43	0.42

Table 85. Results of 64 × 64 dimension images for RGB vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.38	0.41	0.38	0.33
Logistic	0.35	0.34	0.35	0.34
SVM(linear)	0.38	0.39	0.38	0.38
SVM(poly)	0.47	0.48	0.47	0.46
SVM(RBF)	0.56	0.57	0.56	0.55
SVM(sigmoid)	0.08	0.02	0.08	0.03
Random Forest	0.44	0.46	0.44	0.42
Gaussian Naive Bayes	0.37	0.40	0.37	0.36
Decision Tree	0.30	0.31	0.30	0.30
Bernoulli Naive Bayes	0.12	0.20	0.12	0.05
Multinomial Naive Bayes	0.30	0.29	0.30	0.27

Table 86. Results of 64 × 64 dimension images for Grayscale vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.43	0.40	0.43	0.39
Logistic	0.38	0.37	0.38	0.37
SVM(linear)	0.39	0.41	0.39	0.40
SVM(poly)	0.50	0.51	0.50	0.49
SVM(RBF)	0.56	0.58	0.56	0.56
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.46	0.46	0.46	0.44
Gaussian Naive Bayes	0.39	0.42	0.39	0.38
Decision Tree	0.34	0.35	0.34	0.34
Bernoulli Naive Bayes	0.14	0.26	0.14	0.08
Multinomial Naive Bayes	0.31	0.32	0.31	0.30

Table 87. Results of 64 × 64 dimension images for Red channel vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.34	0.44	0.34	0.30
Logistic	0.34	0.34	0.34	0.33
SVM(linear)	0.37	0.38	0.37	0.37
SVM(poly)	0.46	0.47	0.46	0.45
SVM(RBF)	0.54	0.56	0.54	0.54
SVM(sigmoid)	0.08	0.02	0.08	0.03
Random Forest	0.43	0.43	0.43	0.41
Gaussian Naive Bayes	0.36	0.39	0.36	0.35
Decision Tree	0.27	0.28	0.27	0.27
Bernoulli Naive Bayes	0.13	0.30	0.13	0.07
Multinomial Naive Bayes	0.28	0.29	0.28	0.26

Table 88. Results of 64 × 64 dimension images for Green channel vector with Minmax scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.46	0.49	0.46	0.43
Logistic	0.45	0.46	0.45	0.45
SVM(linear)	0.45	0.48	0.45	0.45
SVM(poly)	0.52	0.54	0.52	0.52
SVM(RBF)	0.63	0.63	0.63	0.63
SVM(sigmoid)	0.09	0.02	0.09	0.02
Random Forest	0.51	0.52	0.51	0.47
Gaussian Naive Bayes	0.45	0.51	0.45	0.46
Decision Tree	0.33	0.34	0.33	0.33
Bernoulli Naive Bayes	0.28	0.34	0.28	0.25
Multinomial Naive Bayes	0.41	0.39	0.41	0.39

Table 89. Results of 64 × 64 dimension images for Blue channel vector with Minmax scaling.

7.4.4 Normalizer Scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.41	0.53	0.41	0.38
Logistic	0.52	0.52	0.52	0.50
SVM(linear)	0.57	0.58	0.57	0.57
SVM(poly)	0.74	0.74	0.74	0.74
SVM(RBF)	0.73	0.73	0.73	0.73
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.54	0.58	0.54	0.52
Gaussian Naive Bayes	0.46	0.49	0.46	0.45
Decision Tree	0.40	0.41	0.40	0.40
Bernoulli Naive Bayes	0.28	0.41	0.28	0.24
Multinomial Naive Bayes	0.42	0.43	0.42	0.40

Table 90. Results of 64 × 64 dimension images for RGB vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.33	0.34	0.33	0.29
Logistic	0.38	0.35	0.38	0.35
SVM(linear)	0.37	0.37	0.37	0.35
SVM(poly)	0.51	0.51	0.51	0.51
SVM(RBF)	0.57	0.57	0.57	0.57
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.47	0.45	0.47	0.45
Gaussian Naive Bayes	0.39	0.39	0.39	0.38
Decision Tree	0.29	0.29	0.29	0.29
Bernoulli Naive Bayes	0.11	0.15	0.11	0.03
Multinomial Naive Bayes	0.29	0.29	0.29	0.28

Table 91. Results of 64 × 64 dimension images for Grayscale vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.35	0.36	0.35	0.32
Logistic	0.39	0.36	0.39	0.36
SVM(linear)	0.39	0.37	0.39	0.36
SVM(poly)	0.51	0.51	0.51	0.51
SVM(RBF)	0.57	0.57	0.57	0.57
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.46	0.43	0.46	0.44
Gaussian Naive Bayes	0.39	0.39	0.39	0.37
Decision Tree	0.30	0.31	0.30	0.31
Bernoulli Naive Bayes	0.13	0.27	0.13	0.08
Multinomial Naive Bayes	0.34	0.33	0.34	0.32

Table 92. Results of 64 × 64 dimension images for Red channel vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.29	0.30	0.29	0.25
Logistic	0.38	0.35	0.38	0.35
SVM(linear)	0.38	0.37	0.38	0.36
SVM(poly)	0.50	0.50	0.50	0.50
SVM(RBF)	0.56	0.57	0.56	0.56
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.47	0.48	0.47	0.46
Gaussian Naive Bayes	0.38	0.38	0.38	0.36
Decision Tree	0.27	0.27	0.27	0.27
Bernoulli Naive Bayes	0.13	0.32	0.13	0.07
Multinomial Naive Bayes	0.28	0.28	0.28	0.26

Table 93. Results of 64 × 64 dimension images for Green channel vector with Normalizer scaling.

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.39	0.43	0.39	0.35
Logistic	0.48	0.47	0.48	0.46
SVM(linear)	0.49	0.49	0.49	0.47
SVM(poly)	0.53	0.54	0.53	0.53
SVM(RBF)	0.59	0.59	0.59	0.59
SVM(sigmoid)	0.10	0.01	0.10	0.02
Random Forest	0.50	0.48	0.50	0.47
Gaussian Naive Bayes	0.46	0.46	0.46	0.44
Decision Tree	0.33	0.33	0.33	0.33
Bernoulli Naive Bayes	0.28	0.34	0.28	0.25
Multinomial Naive Bayes	0.39	0.37	0.39	0.36

Table 94. Results of 64 × 64 dimension images for Blue channel vector with Normalizer scaling.

7.5 Results of 128×128 data with standard scaling

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.52	0.57	0.52	0.50
Logistic	0.75	0.75	0.75	0.75
SVM(linear)	0.78	0.78	0.78	0.78
SVM(RBF)	0.85	0.84	0.85	0.84
Random Forest	0.56	0.56	0.56	0.50
Gaussian Naive Bayes	0.50	0.58	0.50	0.51
Decision Tree	0.48	0.48	0.48	0.48

Table 95. Results of 128×128 dimension images with standard scaling on RGB vector

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.44	0.46	0.44	0.39
Logistic	0.40	0.39	0.40	0.39
SVM(linear)	0.47	0.47	0.47	0.46
Random Forest	0.44	0.53	0.44	0.38
Gaussian Naive Bayes	0.36	0.44	0.36	0.34
Decision Tree	0.40	0.40	0.40	0.40

Table 96. Results of 128×128 dimension images with standard scaling on Grayscale vector

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.47	0.46	0.47	0.43
Logistic	0.44	0.43	0.44	0.43
SVM(linear)	0.49	0.49	0.49	0.49
Random Forest	0.47	0.51	0.47	0.41
Gaussian Naive Bayes	0.38	0.46	0.38	0.36
Decision Tree	0.41	0.42	0.41	0.`41

Table 97. Results of 128×128 dimension images with standard scaling on Red channel vector

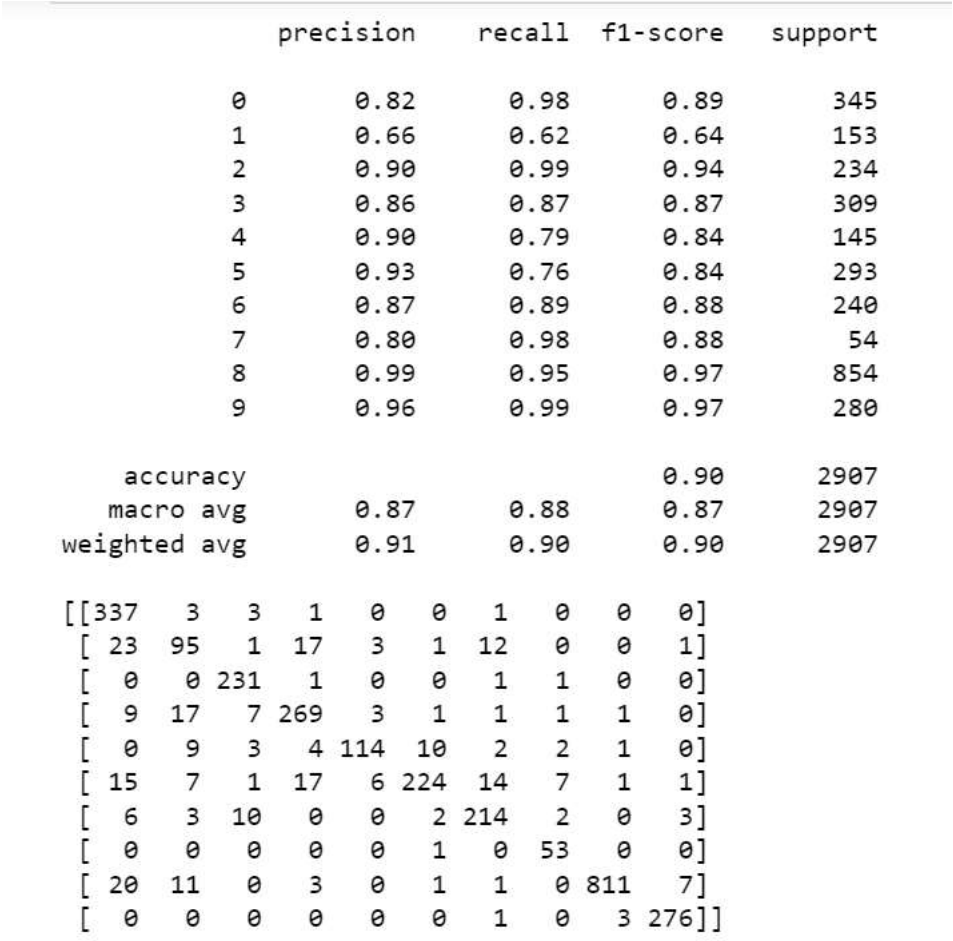
Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.42	0.49	0.42	0.39
Logistic	0.40	0.39	0.40	0.39
SVM(linear)	0.45	0.45	0.45	0.44
Random Forest	0.43	0.43	0.43	0.36
Gaussian Naive Bayes	0.35	0.41	0.35	0.32
Decision Tree	0.41	0.41	0.41	0.41

Table 98. Results of 128×128 dimension images with standard scaling on Green channel vector

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.44	0.46	0.44	0.39
Logistic	0.40	0.39	0.40	0.39
SVM(linear)	0.47	0.47	0.47	0.46
Random Forest	0.44	0.53	0.44	0.38
Gaussian Naive Bayes	0.36	0.44	0.36	0.34
Decision Tree	0.40	0.40	0.40	0.40

Table 99. Results of 128×128 dimension images with standard scaling on Blue channel vector

7.6 CNN Algorithm Results



	precision	recall	f1-score	support
0	0.82	0.98	0.89	345
1	0.66	0.62	0.64	153
2	0.90	0.99	0.94	234
3	0.86	0.87	0.87	309
4	0.90	0.79	0.84	145
5	0.93	0.76	0.84	293
6	0.87	0.89	0.88	240
7	0.80	0.98	0.88	54
8	0.99	0.95	0.97	854
9	0.96	0.99	0.97	280
accuracy			0.90	2907
macro avg	0.87	0.88	0.87	2907
weighted avg	0.91	0.90	0.90	2907

```
[[337   3   3   1   0   0   1   0   0   0]
 [ 23  95   1  17   3   1  12   0   0   1]
 [  0   0 231   1   0   0   1   1   0   0]
 [  9  17   7 269   3   1   1   1   1   0]
 [  0   9   3   4 114  10   2   2   1   0]
 [ 15   7   1  17   6 224  14   7   1   1]
 [  6   3  10   0   0   2 214   2   0   3]
 [  0   0   0   0   0   1   0  53   0   0]
 [ 20  11   0   3   0   1   1   0 811   7]
 [  0   0   0   0   0   0   1   0   3 276]]
```

Figure 28. Results of CNN algorithm on 64×64 images after normalization on RGB vector

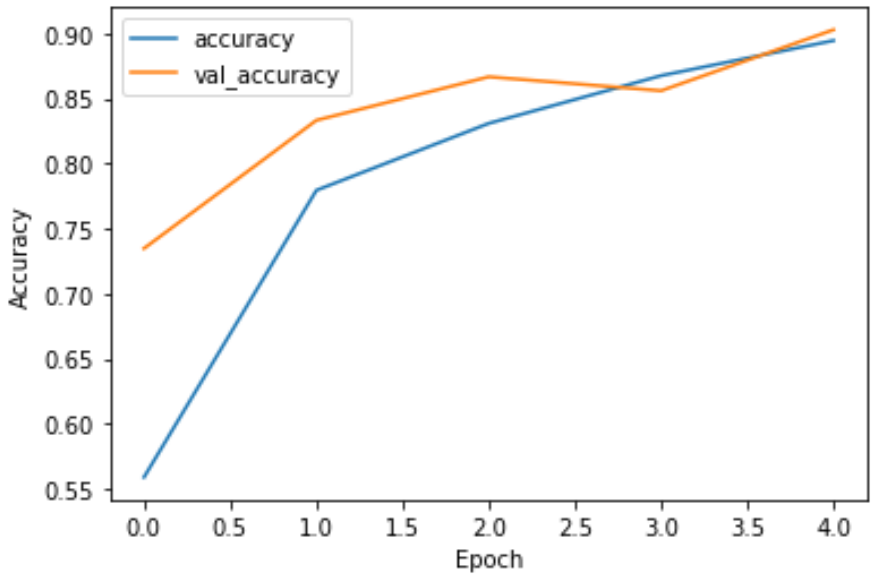


Figure 29. Accuracy vs Epoch graph for CNN model of RGB vector of 64×64 images.

	precision	recall	f1-score	support
0	0.89	0.78	0.83	345
1	0.55	0.49	0.52	153
2	0.73	0.96	0.83	234
3	0.86	0.70	0.77	309
4	0.72	0.64	0.68	145
5	0.79	0.65	0.71	293
6	0.67	0.74	0.70	240
7	0.50	0.76	0.60	54
8	0.90	0.92	0.91	854
9	0.88	0.96	0.92	280
accuracy			0.81	2907
macro avg	0.75	0.76	0.75	2907
weighted avg	0.81	0.81	0.81	2907

```
[[270  17   1   2   0   8   8   2  37   0]
 [  2  75   4   9   6   5  31   7  13   1]
 [  1   0 224   2   1   2   2   1   1   0]
 [  4  25   7 217  14  18  12   3   9   0]
 [  2   3  14   1  93  10   6   8   8   0]
 [ 16  10  13  19   8 191  21  10   5   0]
 [  1   3  41   1   3   4 178   4   5   0]
 [  0   0   3   0   3   4   1  41   2   0]
 [  8   4   1   1   1   1   7   6 789  36]
 [  0   0   0   0   0   0   0   0  11 269]]
```

Figure 30. Results of CNN algorithm on 64×64 images after normalization on Grayscale vector

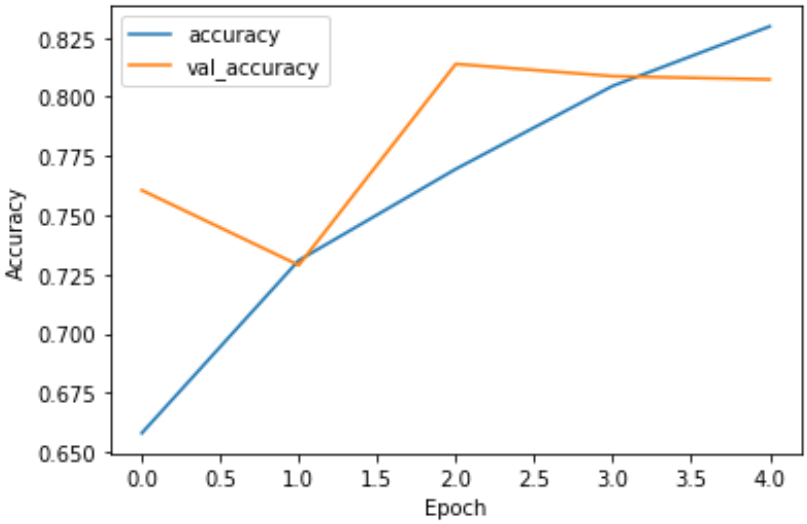


Figure 31. Accuracy vs Epoch graph for CNN model of Grayscale vector of 64×64 images.

	precision	recall	f1-score	support
0	0.76	0.93	0.84	115
1	0.85	0.61	0.71	120
2	0.76	0.98	0.86	125
3	0.88	0.78	0.83	125
4	0.91	0.80	0.85	127
5	0.73	0.73	0.73	117
6	0.85	0.70	0.77	115
7	0.80	0.96	0.87	54
8	0.88	0.91	0.90	114
9	0.79	0.84	0.82	128
accuracy			0.82	1140
macro avg	0.82	0.82	0.82	1140
weighted avg	0.82	0.82	0.81	1140

```
[[107   1   0   1   0   0   0   0   6   0]
 [ 21  73   2   6   3   5   4   0   1   5]
 [  0   0 122   0   0   2   0   0   0   1]
 [  2   6   7  98   2   7   0   0   1   2]
 [  0   2   2   4 101  13   0   2   0   3]
 [  3   2   6   2   5  85   3   3   5   3]
 [  1   0  20   0   0   3  80   1   0  10]
 [  0   0   0   0   0   0   1  52   0   1]
 [  5   0   0   0   0   1   0   0 104   4]
 [  2   2   1   0   0   1   6   7   1 108]]
```

Figure 32. Results of CNN algorithm on 64×64 images after normalization on balanced RGB vector data

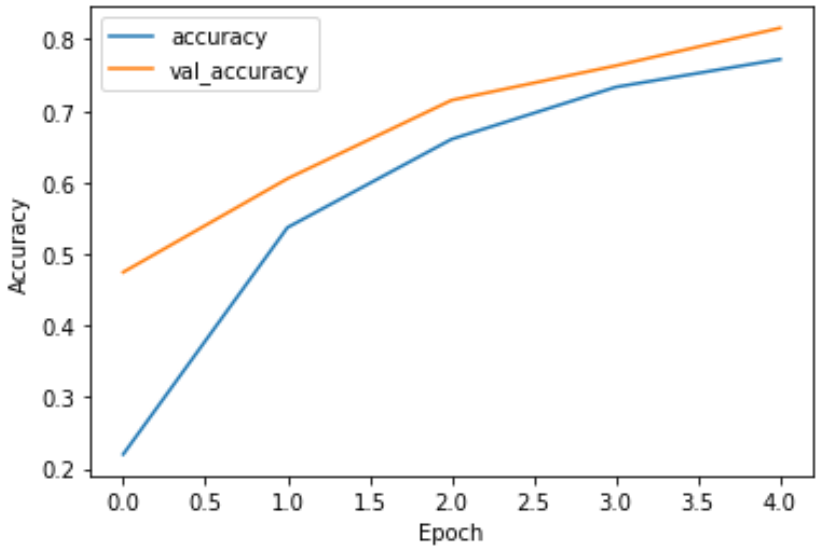


Figure 33. Accuracy vs Epoch graph for CNN model of RGB vector of 64×64 images balanced dataset.

7.7 Comparing the best machine learning models and CNN

7.7.1 Comparing Machine learning models

Initially, the best machine learning model analysis is carried out manually by checking the accuracies of all the models greater than 80 percent. The models with the best accuracy along with the scaling method, vector, accuracy, and algorithm are noted. After the analysis, there are a total of 13 models with more than 80 percent accuracy. Among all the models, non-parametric models performed well than the parametric models on this data. Coming to the vectors, only the RGB vector gives good metrics among all the vectors on all the datasets. No scaling data gave better results than the min-max scaler in the highest models. The algorithm that performed well on all the datasets the is support vector machine (SVM). The SVM kernels which show good results are the RBF and poly. From the above results, we can see that SVM with RBF kernel shows good results than the remaining algorithms. When comparing the best algorithms, There is a similar performance of the poly and RBF kernel after applying the normalizer scaling method to the data. PCA is not performed well on the initial datasets, so it is dropped for the remaining datasets. The best algorithms of each dataset will be discussed further in detail.

When coming to the 32×32 dimensions image dataset, fitting the SVM(RBF) model without applying any scaling gives 84 percent of accuracy. The accuracy increases after applying standard scaling on the features and the other metrics also show better performance than no scaling. The accuracy after applying the standard scaling is 85 percent. The accuracy of the SVM(RBF) on the min-max scaled data is 83 percent. Coming to the normalizer scaled data, SVM with RBF kernel gives an accuracy of 83 percent, and SVM with the poly kernel gives an accuracy of 81 percent. The elbow method shows that 5 components will explain the variance better for this dataset for all the vectors. Later, PCA and ICA are

applied to each vector data and the data is trained using the same process. After applying the PCA and ICA, the computational time for running the model is decreased but the accuracy is decreased. The highest accuracy obtained from the PCA data of the 32×32 dataset is 65 with the SVM algorithm using the RBF kernel and the KNN shows an accuracy of 62 percent.

The 64×64 dataset also shows some similar results to the 32×32 dataset during the analysis. There are more models with better accuracy in this case. Without using any scaling on the data, the SVM algorithm with poly kernel gives an accuracy of 81 percent and the SVM algorithm with RBF kernel gives an accuracy of 85 percent. When coming to the standard scaled data, SVM(RBF) gives an accuracy of 86 percent. Coming to the min-max scaling data, the SVM algorithm shows an accuracy of 80 with poly kernel and 84 percent with the RBF kernel. Interestingly, the SVM algorithm with both poly and RBF kernels shows a nearer accuracy than the normalizer vector. So, the best model is validated using the statistical test and it will be discussed in the validation process. Scree plots of all five vectors show five components are good for analyzing the data. The accuracies of the PCA models show lower metrics than the original dataset. The highest accuracy achieved by the PCA dataset is 65 percent with the SVM algorithm using the RBF kernel.

After the analysis of both datasets, applying PCA is not giving good results with this data. So, the PCA and ICA are not applied to the balanced datasets. So, the results show that the accuracy of the balanced models did not cross 80 percent. When coming to the 32×32 dataset, the maximum accuracy obtained is 77 percent by the SVM algorithm with standard scaled data. The same algorithm gives an accuracy of 76 while using no scaling data.

Balanced dataset of 64×64 also gets good accuracy with the SVM algorithm using the RBF kernel. The accuracy of the data without any scaling is 76 percent and the accuracy is 77 percent after using the standard scaling on the data. These metrics show that the balanced data is also giving good predictions. But due to a lack of data from the source, the research is carried out by taking the first 600 images of each class.

So far, the best model in machine learning is the SVM algorithm using RBF kernel on 64×64 data with standard scaling. The accuracy of the model is 86 percent, and the same model gives 85 percent accuracy with the 32×32 data. This creates a doubt that increasing the dimensions is increasing the accuracy of the model. For checking this assumption, the research is carried out on the 128×128 data with standard scaling. The results show that the SVM algorithm shows 85 percent of accuracy. This shows that the images with 64×64 data are better for analysis of this model. But before confirming the model is better for good predictions we need to compare the other metrics and we need to validate the model. The model show 0.85 precision, 0.86 recall, and 0.85 F1-score. The classification report and the confusion matrix of the best model are shown in the figure 34.

```
                                  precision    recall  f1-score   support

                bacterial_spot       0.88      0.93      0.90       345
                  early_blight       0.64      0.42      0.51       153
                       healthy       0.96      0.95      0.95       234
                   late_blight       0.79      0.81      0.80       309
                     leaf_mold       0.81      0.69      0.74       145
            septoria_leaf_spot       0.79      0.79      0.79       293
                   target_spot       0.79      0.82      0.80       240
          tomato_mosaic_virus        0.75      0.80      0.77        54
tomato_yellow_leaf_curl_virus        0.93      0.96      0.94       854
      two-spotted_spider_mite        0.86      0.86      0.86       280

                      accuracy                           0.86      2907
                     macro avg       0.82      0.80      0.81      2907
                  weighted avg       0.85      0.86      0.85      2907

[[320   2   1   3   0   1   2   0  16   0]
 [ 24  65   2  25   1   4  14   1   9   8]
 [  0   3 222   4   1   2   2   0   0   0]
 [  4   9   3 250  10  10   7   3   7   6]
 [  1   3   0  10 100  19   2   2   5   3]
 [  2   6   1  11  10 232  10   3  15   3]
 [  6   8   2   3   1  10 198   2   2   8]
 [  0   1   0   0   1   8   1  43   0   0]
 [  8   2   0   4   0   5   3   0 822  10]
 [  0   3   0   6   0   2  13   3  11 242]]
```

Figure 34. Classification report and confusion matrix of the best model.

Even though the data has imbalanced classes, it is predicting well on the data. We can see that the testing data is taken based on the classes automatically by the model. This shows that the data is predicted well by the model. The reason for not choosing a balanced dataset is that the data will be reduced to half. Generally, balanced datasets are preferred for predicting the data but in this case, the imbalanced dataset is also predicted well by the model. So, we are taking this for validation.

7.7.2 Comparing CNN models

The CNN algorithm is performed on 64×64 data. The algorithm is performed on the RGB vector of the imbalanced dataset, the grayscale vector of the imbalanced dataset, and the RGB vector of the balanced dataset. The model is run using 5 epochs and the RGB vector of the imbalanced dataset performs well on the data. It achieves an accuracy of 90 and the metrics show that it is predicting well. To know more about the data patterns, the epochs are increased to 10 and rerun the model to know about the data.

	precision	recall	f1-score	support
0	0.96	0.94	0.95	345
1	0.65	0.73	0.69	153
2	0.96	0.97	0.96	234
3	0.92	0.82	0.87	309
4	0.86	0.86	0.86	145
5	0.87	0.86	0.87	293
6	0.91	0.88	0.89	240
7	0.73	0.96	0.83	54
8	0.96	1.00	0.98	854
9	1.00	0.94	0.97	280
accuracy			0.92	2907
macro avg	0.88	0.89	0.89	2907
weighted avg	0.92	0.92	0.92	2907

```
[[323  11   3   2   0   1   2   0   3   0]
 [  6 111   0  11   3   8   7   1   6   0]
 [  0   1 226   2   0   0   4   1   0   0]
 [  2  29   3 254   6  11   0   1   3   0]
 [  0   4   1   2 124   8   1   1   4   0]
 [  1   4   0   4  11 252   5  11   5   0]
 [  2  10   3   1   0   9 210   4   1   0]
 [  0   0   0   0   1   0   1  52   0   0]
 [  2   0   0   0   0   0   0   0 851   1]
 [  0   0   0   0   0   0   0   0  16 264]]
```

Figure 35. Results of CNN model with 10 epochs on RGB vector of 64×64 data.

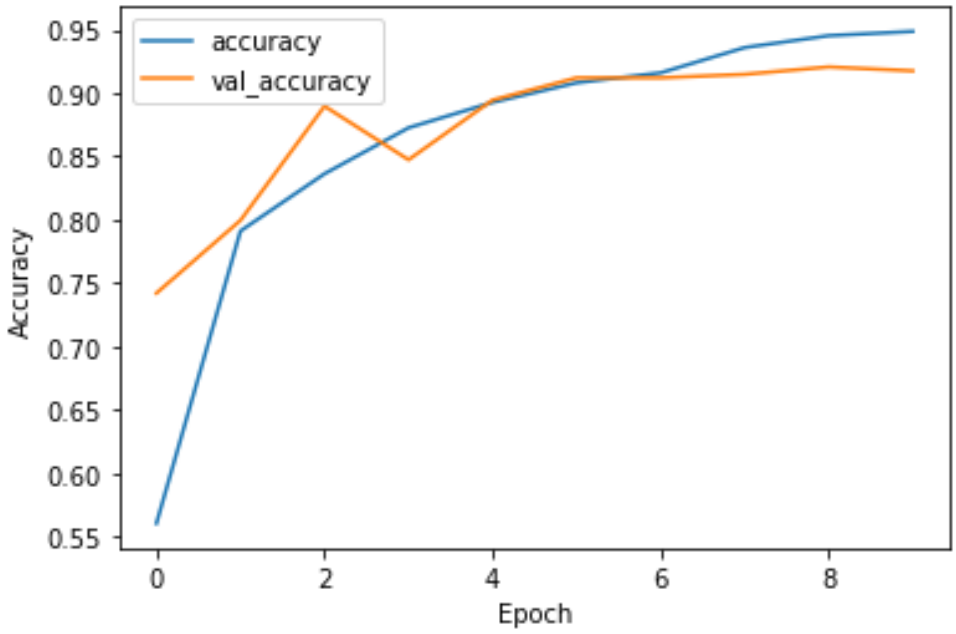


Figure 36. Accuracy vs Epoch graph for CNN model with 10 epochs on RGB vector.

We can see that the deep learning model is giving good accuracy and good metrics after 10 epochs. Normally, deep learning models are used for complex datasets and it needs more training data(Seth DeLand 2019). The dataset has 14,531 images and machine learning can predict the model with 86 percent. So, we can say that if the count of images is increased then it is better to shift to the deep learning models. The accuracy vs validation accuracy of the CNN model shows little distortion at epoch 2 and epoch3. So, the remaining results will be validated using the validation techniques.

7.8. Validation of Machine learning models

7.8.1 Cross-validation and AUC-ROC on the best model

From the above section, we can see that the SVM algorithm is giving noticeable results among all the models. It is important to validate the model before coming to any conclusions. K-fold cross-validation is one of the best ways to know the skill of the model. The data is validated using 8 folds and the accuracy of each fold is examined to know the performance of the model. The data is sorted by the classes, and it needs to be shuffled before doing the process. If the data is not shuffled, then there is a chance of getting folds with the same class data and the model may fail to predict. After cross-validation, the mean of all accuracies gives a mean accuracy of 86.2 percent. The cross-validation scores of each model are shown in the figure 37.

```
scores of all models of k-fold is: [0.84645019 0.86956522 0.87176665 0.86508811 0.84030837 0.87334802
 0.86508811 0.86618943]
```

Figure 37. Accuracies of K-Fold cross-validation of the SVM(RBF) model of standard scaled data.

The accuracies show that the algorithm is performing well on all the folds. The lowest accuracy of a fold is 84 and the highest is 87 percent. So, there are no deviations in the accuracies of the model and it is performing well on all the folds. The model passes the cross-validation method, and it is further analyzed using the AUC-ROC curve. By using the AUC-ROC curve, we can know which class is predicted well by the model. The target value is converted to the binary for analysis and the probabilities of each class are examined. The AUC-ROC curve of the best model is shown in the figure 38.

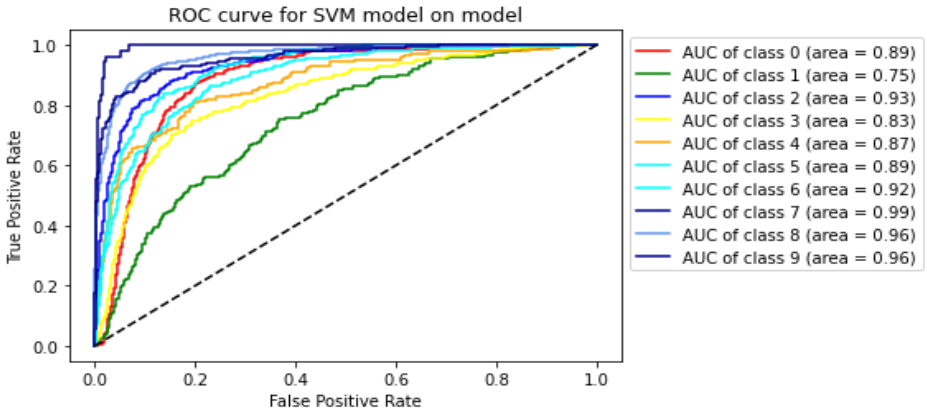


Figure 38. AUC-ROC curve for the SVM model.

The graph shows that the model is better at predicting all the classes. All the classes are showing a good area under the curve value which is greater than 0.70. The less predicted class is the early blight disease which is represented as class 1. The remaining curves show that the model is good at predicting all the classes. The SVM algorithm shows good results in all the validation methods. So, standard scaling is the best scaling method for this dataset, 64×64 dimensions images are better, and SVM with RBF kernel is best for this dataset.

7.8.2 Statistical test for similar models

As discussed in the above section, the SVM algorithm with poly kernel and RBF kernel performs similarly on the normalizer data. A paired statistical t-test is conducted on both models to know which model is better for the normalizer data. The test is dependent because the same data is used for analysis and the mean accuracies of both models will be examined by the statistical test to know which algorithm is performing better. The p-value taken for this test is 0.05. This is performed on the dataset by splitting the dataset into 2 folds and performing the analysis 5 times. In the end, the hypothesis is checked using the p-value. The null hypothesis of the statistical test is "The mean performance of the SVM algorithm with RBF kernel and poly are same" and the alternate hypothesis is "There is a significant difference between the mean performance of the SVM model with RBF kernel and poly kernel". The output of the test is 0.79 and this statistically concludes that there is no difference between the mean performance of the two models. Therefore, we can say there is no difference between using any kernel while using the normalizer scaling on this dataset.

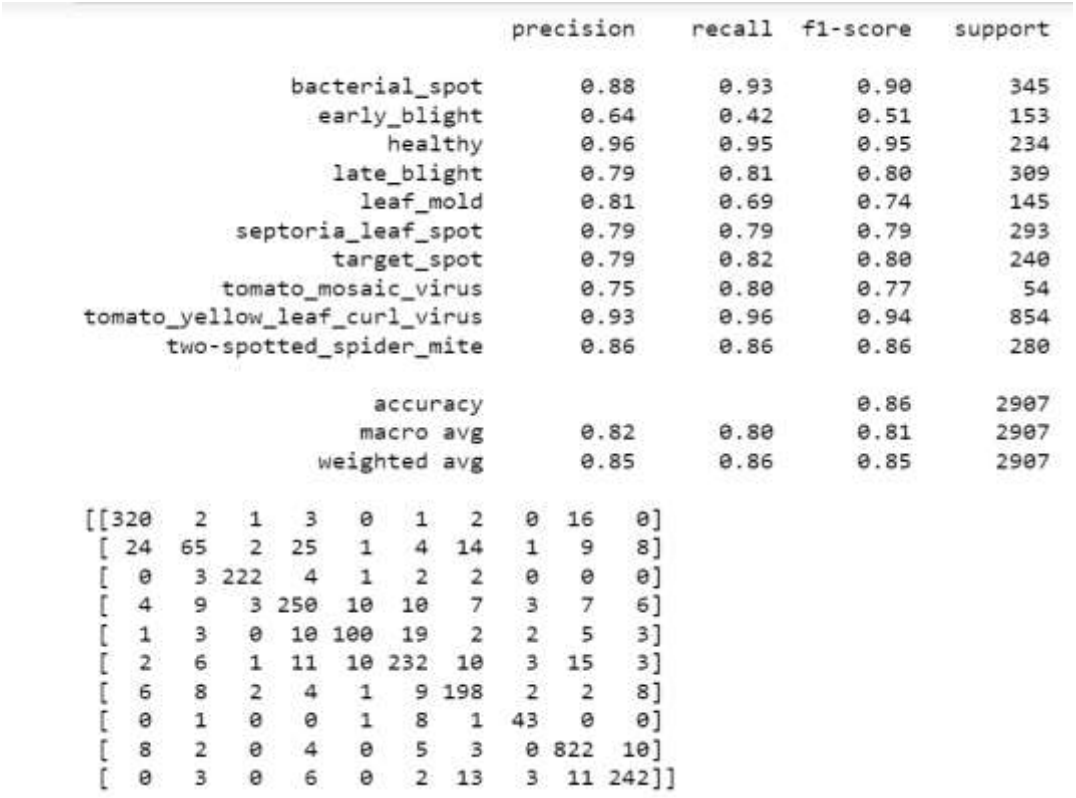
7.9 Adding more features to the best model and checking the classification report

Another research question is to test the data by adding new features to the vectors and check if the accuracy increases. The new features are extracted from the vector data of each image. The new features are the minimum value, maximum value, mean value, and median value of the vector. These columns are added to features and scaled using the standard scaling.

	0	1	2	3	4	5	6	7	8	9	...	12282	12283	12284	12285	12286	12287	mean	min	max	median
0	143	147	146	143	147	146	143	147	146	145	...	120	112	110	123	113	112	97.180664	10	163	101.5
1	178	172	172	177	171	171	175	169	169	173	...	139	125	124	138	124	123	98.775716	5	184	86.0
2	125	129	106	125	129	106	125	128	107	125	...	63	54	55	61	53	51	115.811035	0	219	119.0
3	118	120	117	120	122	119	124	126	123	129	...	180	177	188	184	181	192	99.725260	11	192	89.0
4	166	172	168	166	172	168	166	172	168	167	...	102	109	115	100	107	113	104.349609	23	190	99.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14526	135	120	123	133	118	121	133	118	121	136	...	172	156	169	173	157	170	128.490967	0	197	140.0
14527	131	118	127	131	118	127	132	119	128	134	...	159	141	165	158	140	164	113.974609	2	179	124.0
14528	130	115	118	130	115	118	131	116	119	132	...	129	129	131	127	127	129	124.632568	0	200	129.0
14529	179	168	176	178	167	175	176	165	173	175	...	126	113	120	126	113	122	117.917074	0	182	123.0
14530	114	99	102	112	97	100	109	94	97	108	...	202	188	185	205	191	188	123.845215	0	209	126.0

Figure 39. Data frame after adding mean, minimum, maximum, and median to the existing features.

After adding the features, the data is fitted using the SVM algorithm. The metrics of the data show similar results. The accuracy of the model is 86, precision is 85, recall is 86, and F1-score is 85.



	precision	recall	f1-score	support
bacterial_spot	0.88	0.93	0.90	345
early_blight	0.64	0.42	0.51	153
healthy	0.96	0.95	0.95	234
late_blight	0.79	0.81	0.80	309
leaf_mold	0.81	0.69	0.74	145
septoria_leaf_spot	0.79	0.79	0.79	293
target_spot	0.79	0.82	0.80	240
tomato_mosaic_virus	0.75	0.80	0.77	54
tomato_yellow_leaf_curl_virus	0.93	0.96	0.94	854
two-spotted_spider_mite	0.86	0.86	0.86	280
accuracy			0.86	2907
macro avg	0.82	0.80	0.81	2907
weighted avg	0.85	0.86	0.85	2907

```
[[320   2   1   3   0   1   2   0  16   0]
 [ 24  65   2  25   1   4  14   1   9   8]
 [  0   3 222   4   1   2   2   0   0   0]
 [  4   9   3 250  10  10   7   3   7   6]
 [  1   3   0  10 100  19   2   2   5   3]
 [  2   6   1  11  10 232  10   3  15   3]
 [  6   8   2   4   1   9 198   2   2   8]
 [  0   1   0   0   1   8   1  43   0   0]
 [  8   2   0   4   0   5   3   0 822  10]
 [  0   3   0   6   0   2  13   3  11 242]]
```

Figure 40. Classification report of the new features model.

There are no considerable changes in the output after adding the new features to the data. So, there is no advantage to adding these features to the old features. Almost all the metrics show the same values with the best model. Vector data is enough for machine learning to classify the data. Overall, The results gave answers to all the research questions.

7.10 Deployment

The best model is saved as a pickle file and loaded again for checking how can we use this model in the future. Later, the pickle file is loaded into another program and tested it by giving the input as an image. Before providing a single data as input to the model, the image must be resized to 64×64 size and the data will be converted to the RGB vector. In the next step, we need to place the vector data in a two-dimensional array for predicting the output. Finally, by using the predict function we can get the output of the input. The output of the model is shown in the figure 41.

```
prediction = loaded_model.predict(data2)
```

```
print(prediction)
```

```
['late_blight']
```

Figure 41. Predicting the model by providing input image.

There is no need to provide the entire code to someone for testing the model and there is no need to retrain the model. It can be used by changing the paths in the program. The user needs to edit the path where the sav file exists and the path where the input data is present. We can extend this work by providing a web interface to the model.

8. Conclusion and Future Plan

The priority of this research is to find a machine learning model which is robust and accurate for the prediction. The proposed methodology involves different vectorization methods which are popular and widely used. By following this methodology and the evaluation process, the diseases on the tomato leaf are detected using the image data. Scaling the data increases the accuracy of the model. The best scaling method for this data is a standard scaler and the next best is a normalizer scaler. Min-max scaling is not a preferable option for this dataset. Coming to the vectors, the RGB vector is the best vector for predicting future values and it gives more preferable and noticeable metrics than the other vector techniques. Grayscale vector performed less than the RGB in machine learning as well as in deep learning. Non-parametric models provide better predictions than the parametric models in this analysis. Dimension reduction techniques are not giving good results on the data. Therefore, reducing the data with the PCA and ICA is not a better option for this problem. When comparing the deep learning models, the RGB vector shows better accuracy. This is a multi-classification problem and other metrics need to be validated in the analysis. The SVM algorithm using the RBF kernel shows robust results in all the validation steps and shows a better graph in AUC-ROC also. Coming to the deep learning models, The CNN model provides good output with the RGB vector. The data is not enough for training the data using deep learning models. The results show that CNN achieved 92 percent of accuracy after 10 epochs. When we have a large dataset with complex problems, we can use the deep learning models than the machine learning models for image processing. Answers to all the research questions are provided by the results. In this research, the data is low and there are imbalanced classes of the data. The machine learning models gave a decent accuracy, precision, recall, and F1 score for this dataset. So, we can go with the machine learning model for this problem. The only problem with this research is running time. The system fails to allocate the memory for the features at the analysis stage. For image processing, a system with better GPU and internal RAM storage is needed for increased dimensions. The other option is to take an AWS server for running the model. Google colab free version also failed

to run the model. Due to these problems, the original data with 227×227 dimension images are not explored directly. The deployment of the model shows how the image can be predicted using the prediction function. We can extend the model by using web technologies and hosting a server. This requires some economical support.

The future work of this model is to deploy the model on the web. The model can be deployed on the web by only uploading the sav file. There is no need to provide the training data or vector to the application. The other major advantage is there is no need to train the model in again. The web model needs to take the input from the user from the specified path and convert the data in the back end by resizing the image and converting the data to the RGB vector before testing it. The predicting output will be shown in the front end of the model. By implementing this model, we can provide web services to anywhere in the world. This project can be further improved by providing image data of more diseases. In this research, there are images of ten major tomato leaf diseases and we can predict more diseases if we can get the image data of other diseases. There is a need to follow ethical considerations before adding any new data to the model. If the data is private, then we need to take consent from the source before using the new data. If we get more data and more classes, then we can shift to the deep learning model training and deployment.

9. References

Amrutha K. (2022). *Decision Tree Machine Learning Algorithm - Analytics Vidhya* [online]. Available from: https://www.analyticsvidhya.com/blog/2022/01/decision-tree-machine-learning-algorithm/ [accessed 1 September 2022].

Amy Grant. (2021). *What Causes Tomato Mosaic Virus - Tomato Mosaic Virus Control* [online]. Available from: https://www.gardeningknowhow.com/edible/vegetables/tomato/managing-tomato-mosaic-virus.htm [accessed 12 June 2022].

Aniruddha Bhandari. (2022). *AUC-ROC Curve in Machine Learning Clearly Explained - Analytics Vidhya* [online]. Available from: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/#h2_5 [accessed 21 August 2022].

Ayele, W.Y. (2020). Adapting CRISP-DM for Idea Mining. *International Journal of Advanced Computer Science and Applications*, 11(6).

Badnakhe, M.R. and Deshmukh, P.R. (2011). An Application of K-Means Clustering and Artificial Intelligence in Pattern Recognition for Crop Diseases. , 2011.

Cecilia Lazzaro Blasbalg. (2021). *SWOT Analysis: What Is It and How to Do It for Your Business* [online]. Available from: https://www.wix.com/blog/2019/12/how-to-do-a-swot-analysis/?utm_source=google&utm_medium=cpc&utm_campaign=13774768257^126077909722&experiment_id=^^531699814064^^_DSA&gclid=Cj0KCQjwxtSSBhDYARIsAEn0thQgEAtnl09Zkb6INKGK4L6ZQ6JIdWABU5SOxd684RmeSv2l76oH-WAaAjRzEALw_wcB [accessed 11 June 2022].

Dufaux, F. (2021). Grand Challenges in Image Processing. *Frontiers in Signal Processing* [online], 0, p.3.

Gene McAvoy. (2020). *Take the Right Aim to Tame Target Spot of Tomato - Growing Produce* [online]. Available from: https://www.growingproduce.com/vegetables/take-the-right-aim-to-tame-target-spot-of-tomato/ [accessed 12 June 2022].

Huang, M.-L. and Chang, Y.-H. (2020). Dataset of Tomato Leaves. , 1.

Islam, M., Dinh, A., Wahid, K. and Bhowmik, P. (2017). Detection of potato diseases using image segmentation and multiclass support vector machine. In: *Canadian Conference on Electrical and Computer Engineering*.

Jagtap, S., Bhatt, C., Thik, J. and Rahimifard, S. (2019). Monitoring potato waste in food manufacturing using image processing and internet of things approach. *Sustainability (Switzerland)*, 11(11).

Jason Brownlee. (2020a). *Hypothesis Test for Comparing Machine Learning Algorithms* [online]. Available from: https://machinelearningmastery.com/hypothesis-test-for-comparing-machine-learning-algorithms/ [accessed 22 August 2022].

Jason Brownlee. (2020b). *Parametric and Nonparametric Machine Learning Algorithms* [online]. Available from: https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/ [accessed 20 May 2022].

Khirade, S.D. and Patil, A.B. (2015). Plant disease detection using image processing. *Proceedings - 1st International Conference on Computing, Communication, Control and Automation, ICCUBEA 2015* [online], 13 July 2015, pp.768–771.

Kulkarni, P., Karwande, A., Kolhe, T., Kamble, S., Joshi, A. and Wyawahare, M. (2021). Plant Disease Detection Using Image Processing and Machine Learning. , 2021.

Kurniawati, N.N., Abdullah, S.N.H.S., Abdullah, Salwani and Abdullah, Saad. (2009). Investigation on image processing techniques for diagnosing paddy diseases. *SoCPaR 2009 - Soft Computing and Pattern Recognition* [online], 2009, pp.272–277.

de Luna, R.G., Dadios, E.P. and Bandala, A.A. (2019). Automated Image Capturing System for Deep Learning-based Tomato Plant Leaf Disease Detection and Recognition. *IEEE Region 10 Annual International Conference, Proceedings/TENCON* [online], 2018-October, pp.1414–1419.

Madiwalar, S.C. and Wyawahare, M. v. (2017). Plant disease identification: A comparative study. *2017 International Conference on Data Management, Analytics and Innovation, ICDMAI 2017* [online], 18 October 2017, pp.13–18.

Marjan Kluepfel, James H. Blake, Anthony P. Keinath and Joey Williamson. (2021). *Tomato Diseases & Disorders | Home & Garden Information Center* [online]. Available from: https://hgic.clemson.edu/factsheet/tomato-diseases-disorders/ [accessed 12 June 2022].

Mim, T.T., Sheikh, M.H., Shampa, R.A., Reza, M.S. and Islam, M.S. (2020). Leaves Diseases Detection of Tomato Using Image Processing. *Proceedings of the 2019 8th International Conference on System Modeling and Advancement in Research Trends, SMART 2019* [online], 1 February 2020, pp.244–249.

Nagamani H S and Dr. Sarojadevi H. (2022). Tomato Leaf Disease Detection using Deep Learning Techniques. *IJACSA) International Journal of Advanced Computer Science and Applications* [online], 13(1). Available from: https://thesai.org/Downloads/Volume13No1/Paper_38-Tomato_Leaf_Disease_Detection.pdf [accessed 4 June 2022].

Neetika Khandwlwal. (2022). Image Processing in Python: Algorithms, Tools, and Methods You Should Know - neptune.ai. , 17 January 2022. Available from: https://neptune.ai/blog/image-processing-python [accessed 6 June 2022].

Noah Parsons. (2021). *What Is a SWOT Analysis and How to Do it Right in 2021 (With Examples)* [online]. Available from: https://www.liveplan.com/blog/what-is-a-swot-analysis-and-how-to-do-it-right-with-examples/ [accessed 11 June 2022].

Oishimaya Sen Nag. (2020). *The World's Leading Tomato Producing Countries - WorldAtlas* [online]. Available from: https://www.worldatlas.com/articles/which-are-the-world-s-leading-tomato-producing-countries.html [accessed 12 June 2022].

Patil, Mr.A.N. and Pawar, Miss.V. (2017). Detection and Classification of Plant Leaf Disease. *IARJSET* [online], 4(4), pp.72–75.

Piyush Chaudhary, Anand K. Chaudhari, Dr. A. N. Cheeran and Sharda Godara. (2012). Color Transform Based Approach for Disease Spot Detection on Plant Leaf. *International Journal of Computer Science and Telecommunications* [online], 3. Available from: https://www.ijcst.org/Volume3/Issue6/p12_3_6.pdf [accessed 7 June 2022].

Prateek Bajaj. (2018). *Creating linear kernel SVM in Python - GeeksforGeeks* [online]. Available from: https://www.geeksforgeeks.org/creating-linear-kernel-svm-in-python/ [accessed 1 September 2022].

R. Hazzard. (2013). *Vegetable: Two-spotted Spider Mite | Center for Agriculture, Food, and the Environment at UMass Amherst* [online]. Available from: https://ag.umass.edu/vegetable/fact-sheets/two-spotted-spider-mite [accessed 12 June 2022].

Rajendra Kajale, R. (2015). DETECTION & RECOGNIZATION OF PLANT LEAF DISEASES USING IMAGE PROCESSING AND ANDROID O.S. *International Journal of Engineering Research and General Science* [online], 3(2). Available from: www.ijergs.org [accessed 6 June 2022].

Rangarajan, A.K., Purushothaman, R. and Ramesh, A. (2018). Tomato crop disease classification using pre-trained deep learning algorithm. *Procedia Computer Science* [online], 133, pp.1040–1047.

Salih, T.A., Ali, A.J. and Ahmed, M.N. (2020). Deep Learning Convolution Neural Network to Detect and Classify Tomato Plant Leaf Diseases. *OALib*, 07(05).

Sandeep Balachandran. (2020). *Machine Learning - Going Furthur with CNN Part 2 - DEV Community* [online]. Available from: https://dev.to/sandeepbalachandran/machine-learning-going-furthur-with-cnn-part-2-41km [accessed 7 June 2022].

Sanjay, P., Dhaygude, B., Nitin, M. and Kumbhar, P. (2013). Agricultural plant Leaf Disease Detection Using Image Processing. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* [online], 2(1). Available from: www.ijareeie.com [accessed 6 June 2022].

Sanjay Patil. (2011). Leaf disease severity measurement using image processing. *International Journal of Engineering and Technology* [online], 2011. Available from: https://www.researchgate.net/publication/267430095_Leaf_disease_severity_measurement_using_image_processing [accessed 6 June 2022].

Seth DeLand. (2019). *When to use Machine Learning or Deep Learning? - Embedded Computing Design* [online]. Available from: https://embeddedcomputing.com/technology/ai-machine-learning/when-to-use-machine-learning-or-deep-learning [accessed 24 August 2022].

Sharma, P., Hans, P. and Gupta, S.C. (2020). Classification of plant leaf diseases using machine learning and image preprocessing techniques. *Proceedings of the Confluence 2020 - 10th International Conference on Cloud Computing, Data Science and Engineering* [online], 1 January 2020, pp.480–484.

Singh, A. and Kaur, H. (2021). Potato Plant Leaves Disease Detection and Classification using Machine Learning Methodologies. , 2021.

Singh, V. and Misra, A.K. (2017). Detection of plant leaf diseases using image segmentation and soft computing techniques. *Information Processing in Agriculture* [online], 4(1), pp.41–49.

Tony Ylu. (2019). *Understanding Random Forest. How the Algorithm Works and Why it Is… | by Tony Yiu | Towards Data Science* [online]. Available from: https://towardsdatascience.com/understanding-random-forest-58381e0602d2 [accessed 1 September 2022].

Trivedi, N.K., Gautam, V., Anand, A., Aljahdali, H.M., Villar, S.G., Anand, D., Goyal, N. and Kadry, S. (2021). Early detection and classification of tomato leaf disease using high-performance deep neural network. *Sensors*, 21(23).

Vetal, S. and R.S., K. (2017). Tomato Plant Disease Detection using Image Processing. *IJARCCE* [online], 6(6), pp.293–297.

What is the k-nearest neighbors algorithm? | IBM. Available from: https://www.ibm.com/topics/knn [accessed 1 September 2022].

Zhang, S.W., Shang, Y.J. and Wang, L. (2015). PLANT DISEASE RECOGNITION BASED ON PLANT LEAF IMAGE. *J. Anim. Plant Sci* [online], 25(3).

Appendix