**Interim Report**

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

**Project Title:**

**Researcher Name:** S. V. M. Ravichandra Reddy Kovvuri

**Researcher Email:** [D00251806@student.dkit.ie](mailto:D00251806@student.dkit.ie)

**Research Supervisor:** Dr. Abhishek Kaushik

**Course:** Master’s in Data Analytics

**Department:** Computer Science and Mathematics

**Institution:** Dundalk Institute of Technology

**Academic Year:** 2021 - 2022

**Acknowledgment**

This study is not supported by any organization. This is intended for academic and learning purposes. 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.

**Table of Contents** Page No.

Title page …………………………………………………………………..1

Acknowledgment…………………………………………………………...2

Table of Contents …………………………………………………………..3

Abstract …………………………………………………………………….4

1. Introduction ……………………………………………………………...4

1.1 Research Objectives ………………………………………………….5

1.2 Research Questions …………………………………………………..5

2. Literature Review ………………………………………………………..6

2.1 Tools Required and Common Problems ……………………………..6

2.2 Machine Learning Models …………………………………………...6

2.3 Deep Learning Models ……………………………………………….8

2.4 Understanding from Related Works ………………………………...10

3. Methodology …………………………………………………………….10

3.1 Data Collection ...................................................................................11

3.2 Context Understanding ……………………………………………...12

3.3 Data Understanding ............................................................................12

3.4 Data Preparation .................................................................................13

3.5 Modeling .............................................................................................14

3.6 Evaluation ...........................................................................................14

3.7 Deployment .........................................................................................14

4. Data Pre-processing ....................................................................................15

5. Ethical Considerations and SWOT Analysis ..............................................19

5.1 Ethical Considerations ……………………………………………….19

5.2 SWOT Analysis ...................................................................................20

6. Evaluation Metrics ……….……………………………………………….21

6.1 Evaluation ............................................................................................21

7. Results …………………………………………………………………….21

8. Conclusion and Future Plan ………………………………………….21

7. References ………………………………………………………………...22

8. Appendix ………………………………………………………………….25

***Abstract –* Food security is a major concern in any country. Farmers face many problems while cultivating the 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 Tomato, Potato, and Corn. Reduction in agriculture production will rise the imports and this will change many factors in the country. Advancements in the agriculture field are much needed and some poor nations still working with old methods. Lack of knowledge in cultivation and diseases is a major issue for many farmers. Most of the farmers implement traditional farming methods and they are unable to detect the diseases in the early stage. Machine learning and deep learning techniques are capable of detecting the diseases at early stages and help the farmers to take precautions against the disease. 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.

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 the healthy leaves. So, the final model can be capable to find these nine diseases and healthy plants.

An image is a collection of different pixels and the values of these pixels decide the colour of the pixel. The images must be converted to different 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. 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 original data from the contributor, data with a maximum count of 600 in each category, 32×32-dimension image data, and 64×64-dimension image data. These four different types of analysis are explained using Table 1.

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| 128×128 | Data with 128×128-dimension images |
| Maximum count 600 | Data with a maximum count of 600 in each category |
| 32×32 | Data with 32×32-dimension images |
| 64×64 | Data with 64×64-dimension images |

Table .1 Data used in this study

Helping farmers and motivating them to use the new technologies is the aim of this project. 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 the 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?

**2. Literature Review**

Image processing is a trending and interesting technology to work. 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 the 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) 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 the 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 the 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 the 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 the 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 machine 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 the 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 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 the 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 the 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 accuracies 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 CNN, KNN, and SVM.

**3. Methodology**

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 the 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. Other advantages of this methodology are flexibility and dependencies. 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. The methodology of this project is explained in Figure 1.

Data

Collection

Figure 1. Methodology of the project

**3.1 Data Collection**

In the data collection phase, the primary data needed for this research is the images. So, the best way to collect data is by taking data from the internet. 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 hygiene. 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. After searching and exploring different related works and datasets related to the tomato leaves images, there is a website named Mendeley 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, hygiene, and relevant to the tomato leaves. The ethical considerations for 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, the 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. There is a need for different python libraries for converting the images to different vectorization models and performing the machine learning algorithms. Python programming language is used for the overall analysis of the data because it is one of the popular and robust languages for data analysis. For handling the data different python libraries are needed and the major libraries are Pandas and NumPy. And Pillow library can be used for the conversion of the images to vector forms. Finally, the sci-kit learn library is needed for analyzing the data using machine learning techniques. 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 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 are shown in Figure 2.

A green leaf on a white surface

Description automatically generated with medium confidence A picture containing custard apple

Description automatically generated A picture containing ground

Description automatically generated A picture containing lettuce, mildew, vegetable

Description automatically generated A picture containing vegetable

Description automatically generated

Bacterial Spot Early Blight Late Blight Leaf Mold Septoria Leaf Spot

A green leaf on a grey surface

Description automatically generated with low confidence A green leaf on a grey surface

Description automatically generated with low confidence A green leaf on a grey surface

Description automatically generated with medium confidence A green leaf on a grey surface

Description automatically generated with medium confidence A green leaf on a grey surface

Description automatically generated with medium confidence

Target Spot Mosaic Virus Yellow Leaf Curl Two-spotted Healthy

Virus Spider Mite leaf

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

**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, these vector values of all pixels in each image are extracted and saved in the NumPy array. 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. These vector values can be extracted using functions from the pillow library. The images are opened in the python library 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 each for the five vector methods. The same process will be repeated by resizing the images to 32×32 size and 64×64 size. 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**

In the modeling phase, The converted data from the preparation phase is used to analyze the prediction by implementing different machine learning algorithms. The idea 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. Since this is a classification problem, algorithms that are popular and used in related works like Logistic, SVM, and KNN classifiers will be 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 and other metrics of the different models. In the later stages, the data after doing feature scaling will be analyzed. Different scaling methods will be also performed on the data and observe which method will be giving good prediction.

**3.6 Evaluation**

Evaluation of the models is not an easy task. Looking at the accuracy alone will not give the best analysis 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 70 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 help in choosing the best methods 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.

**3.7 Deployment**

Finally, the model which shows good metrics will be taken for the deployment. It is better to pick the best methods and train the method for deployment to ensure the performance of the code. The major goal of the deployment is to evaluate the context of the model. After the deployment, we can examine how the model is working and how it is predicting the diseases of the tomato leaves. Building the model again with the best methods will increase the efficiency of the model. Once the model is deployed we can use it for the analysis as many times as possible. If the model fails in the deployment phase in terms of context then the entire process should be re-run again.

**4. Data Pre-processing**

Images are understandable to humans and normally people will get more information from an image. This is not the case for analyzing the images for the machines. 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. 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 227\*227 size. 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 pre-processing are explained in Figure 2.

Images placed with correct folder names of their diseases

Resizing Images to different sizes and save to different Folders

To check the count bias: Images with a maximum count of 600 is taken for all the diseases

Converting images to different vector formats and saving it in a NumPy array

Saving the features and disease in a CSV file

Figure 3. Data Pre-processing

The images are already separated using folders based on the disease type by the contributor. The name of the folder contains the disease name and some numbers. So, the first step in pre-processing is to save the folders with the correct name. Later the images are resized to 64\*64 and 32\*32 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.

Text

Description automatically generated

Figure 4. Data stored in different folders based on the type

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. 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.

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

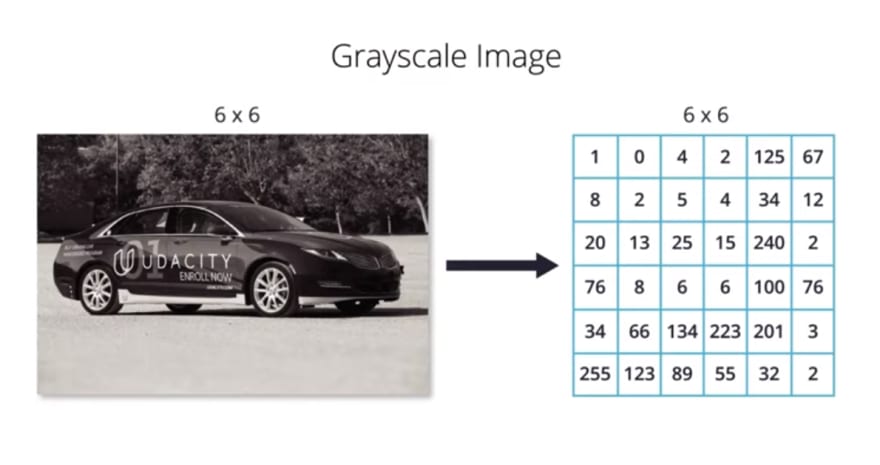


Figure 5. grayscale to vector. Source: (Sandeep Balachandran 2020)

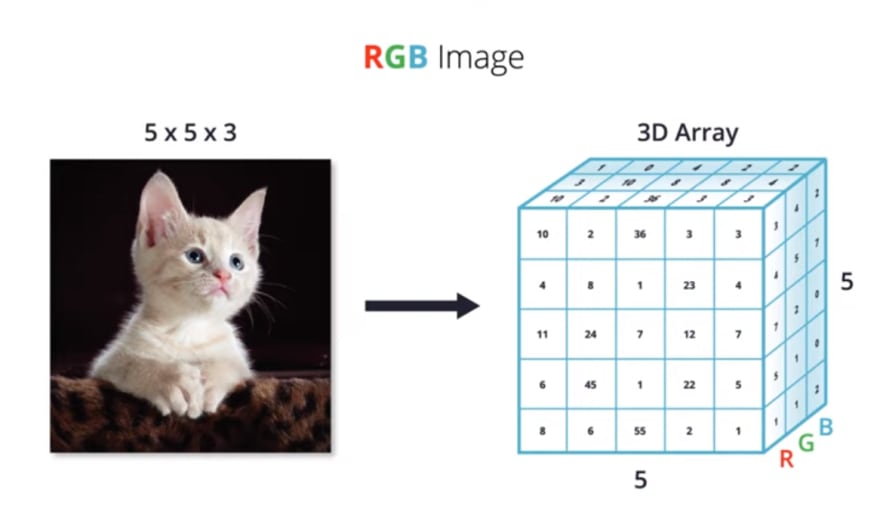


Figure 6. RGB image to vector format. Source: (Sandeep Balachandran 2020)

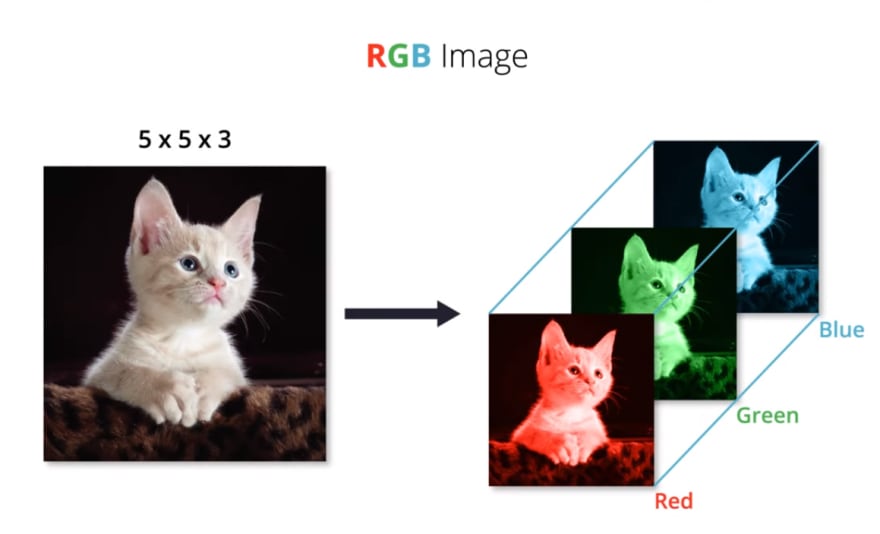


Figure 7. Image to RGB vector format. Source : (Sandeep Balachandran 2020)

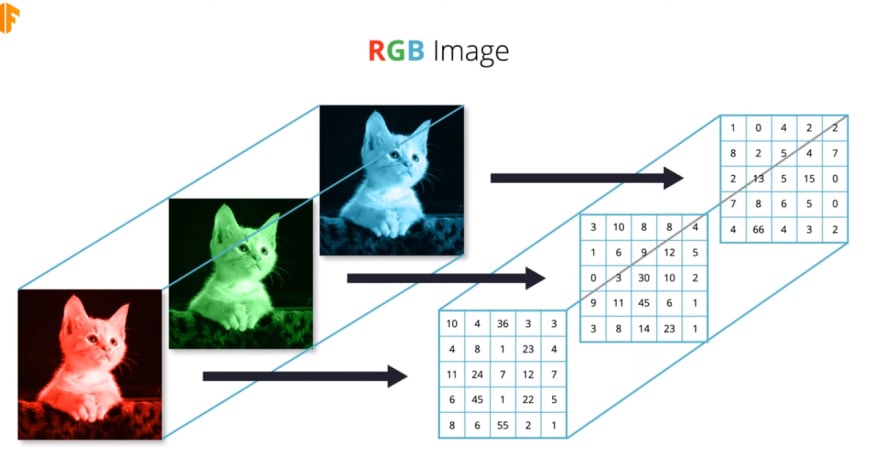


Figure 8. RGB to individual vector format. Source : (Sandeep Balachandran 2020)

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 the machine learning algorithms in 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.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 9. data of different vector forms

The below Table 2 shows how the data is saved in all the folders after Pre-processing:

|  |  |
| --- | --- |
| Folder | Sub-Folders and data available |
| 32 | data(images with dimension 32×32), vector(.npy and CSV files of images in data), no\_bias(images with maximum count 600 of each disease), no\_bias\_vector(.npy and CSV files of no\_bias folder images). |
| 64 | data(images with dimension 64×64), vector(.npy and CSV files of images in data), no\_bias(images with maximum count 600 of each disease), no\_bias\_vector(.npy and CSV files of no\_bias folder images). |
| no\_bias | data(contains images with a maximum count of 600 for each disease with dimension 227×227). |
| Preprocessed\_data | data(images from contributor), vector(.npy and CSV files of images in data folder). |

Table 2. Data stored in the system after pre-processing

**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, Timesaving, 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 tomato 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). The SWOT model for this project is shown in Table 3.

|  |  |
| --- | --- |
| **Strengths**  1. Cost-effective: this can be implemented with less amount.  2. Accuracy: Normally, image processing models give good accuracy.  3. Time reduction: This will reduce the time taken for examining the disease.  4. Flexible vectors: The vector information is stored in the form of lists. So, this data less time to process the information. | **Weakness**  1. Time-consuming: Takes more time for converting the data to vector form.  2. Noise Effect: If there is noise in the images then it is not capable to predict correctly.  3. Reach financials: This model is built intended to help farmers. So, this needs financials to educate farmers about this model. |
| **Opportunities**  1. Extendable: This model can be used for other crops also.  2. Geographical: More than 150 countries farm tomatoes.  3. Partnership: Fertilizer companies invest in these types of technologies.  4. Trending technology: Image processing is a reliable and attractive technology. | **Threats**  1. New Technologies: Deep Learning provides more accurate predictions if more data is provided.  2. Services: Requires lots of services according to the crop. |

Table 3. SWOT Analysis of this project

**6. Evaluation Metrics**

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. In the practical usage of the model the input data may increase, or we need to add some more diseases for predictions. The model needs a plan to meet future expectations.

**6.1 Evaluation**

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 =
2. Precision =
3. Recall =
4. F1 Score =

Firstly, the models which give more than 70 percent accuracy will be selected and moved to further analysis. Later, the classification report and other metrics of each model will be examined carefully. Generally, a classification report is a good method for evaluating multi-classification problems. The models which give a good score in the classification report will be moved to further analysis. Lastly, the model which shows reasonable and robust results in all the metrics will be selected and moved forwarded to the deployment phase.

**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** |
| 7.1 | 32 × 32 | Without Scaling,  Standard Scaling,  Minmax scaling,  Normalizer scaling,  PCA |
| 7.2 | 64 × 64 | Without Scaling,  Standard Scaling,  Minmax scaling,  Normalizer scaling,  PCA |
| 7.3 | Balanced data  (32 × 32) | Without Scaling,  Standard Scaling,  Minmax scaling,  Normalizer scaling |
| 7.4 | Balanced data  (64 × 64) | Without Scaling,  Standard Scaling,  Minmax scaling,  Normalizer scaling |
| 7.5 | 128 × 128 | Without Scaling,  Standard Scaling,  Minmax scaling,  Normalizer scaling,  PCA |
| 7.6 | CNN algorithm | Normalised data |
| 7.7 | Comparing best machine learning model and CNN | Best scaling method from Machine learning |

Table 4. Results section structure

The weighted average of the metrics is taken into consideration for the selection because the count of the data is not equal in all the classes. The best model from the metrics will be compared with the deep learning model at the end of the results section.

**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 | 0.46 | 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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 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.

**PCA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

**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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 Balanced in count (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 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 | 0.28 | 0.41 | 0.28 | 0.24 |

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. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |

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. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| KNN | 0. | 0. | 0. | 0. |
| Logistic | 0. | 0. | 0. | 0. |
| SVM(linear) | 0. | 0. | 0. | 0. |
| SVM(poly) | 0. | 0. | 0. | 0. |
| SVM(rbf) | 0. | 0. | 0. | 0. |
| SVM(sigmoid) | 0. | 0. | 0. | 0. |
| Random Forest | 0. | 0. | 0. | 0. |
| Gaussian Naive Bayes | 0. | 0. | 0. | 0. |
| Decision Tree | 0. | 0. | 0. | 0. |
| Bernoulli Naive bayes | 0. | 0. | 0. | 0. |
| Multinomial Naive bayes | 0. | 0. | 0. | 0. |

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

**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 we can able to predict the disease on the tomato leaves. Machine learning algorithms give worthy solutions to many problems, and it provides essential for many fields. But the efficiency of machine learning models decreases with the increase in memory so Deep learning is an alternative in these cases. After finding a suitable and reasonable machine learning algorithm, we will explore the popular deep learning algorithms like CNN for further analysis.

**7. References**

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].

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).

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].

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 a 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].

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.

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.

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