TOMATO LEAF DISEASE DETECTION

A Course Project report submitted

in partial fulfilment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

by

D. SANJAY SWARAN KUMAR	**(2103A52013)**
G. RAVALIKA	**(2103A52158)**
K. MADHUSRI	**(2103A52090)**

Under the guidance of

Mr. Ramesh

Assistant Professor, Department of CSE.

SR UNIVERSITY

Department of Computer Science and Artificial Intelligence

SR UNIVERSITY

Department of Computer Science and Artificial Intelligence

CERTIFICATE

This is to certify that project entitled **“TOMATO LEAF DISEASE DETECTION”** is the bonafied work carried out by **D. Sanjay Swaran Kumar, G. Ravalika, K. Madhusri** as a Course Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** during the academic year2022-2023 under our guidance and Supervision.

Mr. Ramesh	**Dr. M.Sheshikala**
Asst. Professor,	Assoc. Prof. & HOD
(CSE),	
S R University,	S R University,
Ananthasagar, Warangal.	Ananthasagar, Warangal.

ACKNOWLEDGEMENT

We express our thanks to Course co-coordinator **Mr. Ramesh, Asst. Prof.** for guiding us from the beginning through the end of the Course Project. We express our gratitude to Head of the department CS&AI, **Dr. M.Sheshikala, Associate Professor** for encouragement, support and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dean, School of Computer Science and Artificial Intelligence, **Dr C. V. Guru Rao**, for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

Plant phenotyping is a critical aspect of characterizing plant for plant growth monitoring. This paper introduces an efficient approach to identify healthy and diseased or an infected leaf using image processing and machine learning techniques. Various diseases damage the chlorophyll of leaves and affect with brown or black marks on the leaf area. These can be detected using image preprocessing, image segmentation, feature extraction, and classification using machine learning algorithms. For feature extraction, Grey Level Co-occurrence Matrix (GLCM) is applied. Support Vector Machine (SVM) is one of the machine learning algorithms is used for classification. The Convolutional Neural Network (CNN) resulted in a improved accuracy of recognition compared to svm approach…

Table of Contents

1.INTRODUCTION

1.1. Overview

Agriculture production in the Indian economy is more than just food. Today's agricultural land mass hasgrown so large that it has become an important part of its economy. In India, 60-70% of population relies on agriculture sector. Plant diseases often cause severe loss of vegetables and crops. Plant diseasescan also affect human health by secreting toxic metabolites. The study of plant disease involves detection of visual patterns in the plants. Diagnosis of plant diseases is an important part of cultivation as failure will affect the quantity and quality of product and human health .

There are various types of plant diseases caused by organisms like virus, bacteria and fungus. An automated disease identification process can be helpful in identifying plant pathology at an early stage. The early detection of disease has a positive effect on plant health. In most of the cases, disease symptoms are seen on the leaves, stem and fruit. The indications on the plant leaves are used to diagnosethe disease faster, more reliably and at lower costs. In general, the technique for diagnosing plant diseases is naked eye inspection by farmers, which allows for disease recognition and detection.

A large number of specialists and constant plant monitoring is required for this, which incur a cost whendealing with large farms. However, in certain nations, farmers lack adequate facilities or even the knowledge of how to contact experts. This means that consulting professionals is both expensive and time consuming. Image processing and machine learning methods were also used to identify various plant diseases before the emergence of deep learning. To prepare images for the next steps, image processing methods such as image enhancement, segmentation, colour space conversion, and altering areused. The image's key features are then extracted and used as the input for the classifier.

The overall classification precision is determined by the image processing and feature extraction techniques. However, recent research has shown that networks trained on generic data will achieve state-of-the-art efficiency. CNNs are supervised multi-layer networks that can dynamically learn features from datasets. In almost all significant classification tasks, CNNs have recently achieved state-of-the-art results. In the same architecture, it will isolate features and classify them.

1.2. Problem Statement:

Plant diseases have turned into a dilemma as it can cause significant reduction in both quality andquantity of agricultural products.

To make an efficient use of Machine Learning Algorithms which reduces time and cost for Farmer todetect the plant disease, its effect on crop yield and suggest the pesticides for Plant disease.

1.3. Existing solution

The existing method for plant disease detection is simply naked eye observation by experts through which identification and detection of plant diseases is done. For doing so, a large team of experts as well as continuous monitoring of plant is required, which costs very high when we do with large farms. At the same time, in some countries, farmers do not have proper facilities or even idea that they can contact to experts. Due to which consulting experts even cost high as well as time consuming too. In such conditions, the suggested technique proves to be beneficial in monitoring large fields of crops. Automatic detection of the diseases by just seeing the symptoms on the plant leaves makes it easier as well as cheaper. This also supports machine vision to provide image based automatic process control, inspection, and robot guidance.

Plant disease identification by visual way is more laborious task and at the same time, less accurate and can be done only in limited areas. Whereas if automatic detection technique is used it will take less efforts, less time and become more accurate. In plants, some general diseases seen are brown and yellow spots, early and late scorch, and others are fungal, viral and bacterial diseases. Image processing is used for measuring affected area of disease and to determine the difference in the color of the affected area.

1.4. PROPOSED SYSTEM

A proposed system for sculpture detection would leverage deep learning-based approaches to automatically identify and analyze sculptures in images or videos. The proposed system could include the following components:

- **Data collection**: The system would need a large dataset of images or videos of sculptures to train the deep learning models. This data could be collected from various sources, such as museum collections, public art installations, and online image repositories.

- **Preprocessing**: The images or videos in the dataset would need to be preprocessed to extract features such as color, texture, and shape. This could involve techniques such as image segmentation and feature extraction.

- **Training**: The system would use deep learning models, such as Convolutional Neural Networks (CNNs), to learn features from the preprocessed images or videos. The models would be trained on a labeled dataset, where each sculpture is labeled with its corresponding class.

- **Testing and evaluation**: Once the models are trained, they would be tested on a separate set of images or videos to evaluate their performance. The performance could be evaluated using metrics such as precision, recall, and F1 score.

- **Deployment**: The trained models could be deployed in a software application that allows users to upload images or videos and automatically detect and classify sculptures. The application could also provide additional information about each sculpture, such as its history, artist, and location.

Overall, a proposed system for sculpture detection would leverage deep learning models to automate the process of identifying and analyzing sculptures in images or videos. This would significantly reduce the time and labor required for manual analysis, and enable art experts to study and preserve sculptures more efficiently and accurately.

LITERATURE SURVEY

Reference Number	**Title**	**Machine Learning Algorith m**	**Accuracy**	**Author**
[1]	Plant Disease Detection using Deep Learning	Convolutional Neuralnetwork	98.3%	Rozina Chohan & Murk Chohan
[2]	Using deep learning for Image-based plant disease detection	Deep Convolutional Neural network	98.21%	Sharada P.Mohanty & David P.Hughes
[3]	Plant Leaf diseas e detecti on and classif ication using Machine Learning	Gray co-occurrenceMatrix method and Support Vector machine	Minimum:95.774% and Maximum:98.874 %	Vijeta Shrivastava & Indrajit Das

[4]	Deep learning models for plant Leaf disease detection	Convolutional Neural network	99.53%	K P. Fereninos
[5]	Plant disease Identification based on Deep learning algorithm in Smart Farming	Region Proposal Network (RPN) algorithm and CV algorithm	83.57%	Yan Guo & Jin Zhang
[6]	Solving Current limitations of deep learning based approaches for plant leaf disease detection	Traditional Augmentation methods , GAN and neural networks.	93.47%	Marko Arsenovic & Mirjana Karanovi
[7]	Applicatio n of Random Forest for Classificati on of Grapes Diseases from Images Captured in Uncontrolled Environments	Probabilistic Neural Network (PNN) ,Back Propagation Neural Network (BPNN) ,Support Vector Machine (SVM) and Random Forest	86%	Sandika Biswas, Avil Saunshi & Sanat Sarangi

[8]	Intelligent Plant Disease Identification System Using Machine Learning	Extreme Learning Machine (ELM) and Support Vector Machine (SVM) with linear and polynomial kernels	95%	Paramasivam Alagumariappan, Najumnissa Jamal Dewan

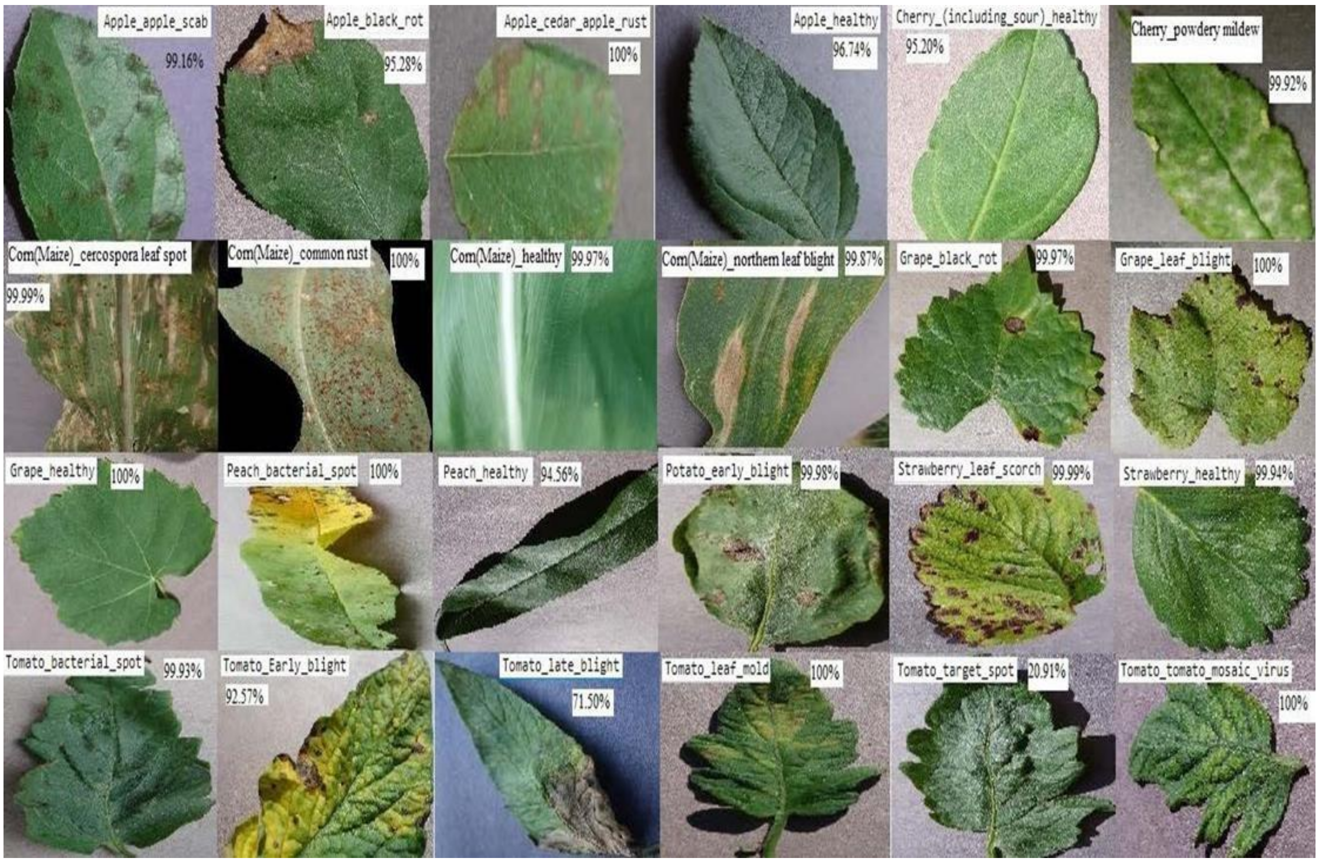
DATA PREPROCESSING

This data set contains 1000 images, which were then classified into 2 classes: Healthy and Disease(1,0)

3.1. Dataset Description

- The dataset consists of 500 Healthy images of Tomato Leaf and 500 Disease images of Tomato Leaf.

Some pictures of diseased leaf:


Apple_apple_scab
99.16%
Apple_black_rot
95.28%
Apple_cedar_apple_rust
100%
Apple_healthy
96.74%
Cherry_(including_sour)_healthy
95.20%
Cherry_powdery mildew
99.92%
Corn(Maize)_cercospora leaf spot
99.99%
Corn(Maize)_common rust
100%
Corn(Maize)_healthy
99.97%
Corn(Maize)_northern leaf blight
99.87%
Grape_black_rot
99.97%
Grape_leaf_blight
100%
Grape_healthy
100%
Peach_bacterial_spot
100%
Peach_healthy
94.56%
Potato_early_blight
99.98%
Strawberry_leaf_scorch
99.99%
Strawberry_healthy
99.94%
Tomato_bacterial_spot
99.93%
Tomato_Early_blight
92.57%
Tomato_late_blight
71.50%
Tomato_leaf_mold
100%
Tomato_target_spot
20.91%
Tomato_tomato_mosaic_virus
100%

3.2. Data Cleaning

Data cleaning is an important step in any machine learning task, including car object detection. It involves identifying and correcting errors and inconsistencies in the dataset to ensure that the data is accurate, complete, and ready for analysis.

Here are some common data cleaning steps that can be applied to sculpture detection datasets:

Remove duplicates: Duplicates can occur when multiple images of the same scene are captured. Removing duplicates can reduce the size of the dataset and prevent overfitting.

Remove outliers: Outliers can occur when the dataset contains images that do not represent the typical distribution of the data. Removing outliers can improve the accuracy of the model and prevent it from being biased towards non-representative data.

Check for missing data: Missing data can occur when some of the images in the dataset do not contain the relevant information. Checking for missing data and either removing or imputing missing values can improve the accuracy of the model.

Normalizing:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
d= sc_x.fit_transform(d)#normalizing
e = sc_x.transform(e)
```

Normalize the data: Normalizing the data involves scaling the pixel values of the images to a standard range, which can improve the performance of the model and reduce the impact of lighting and color variations.

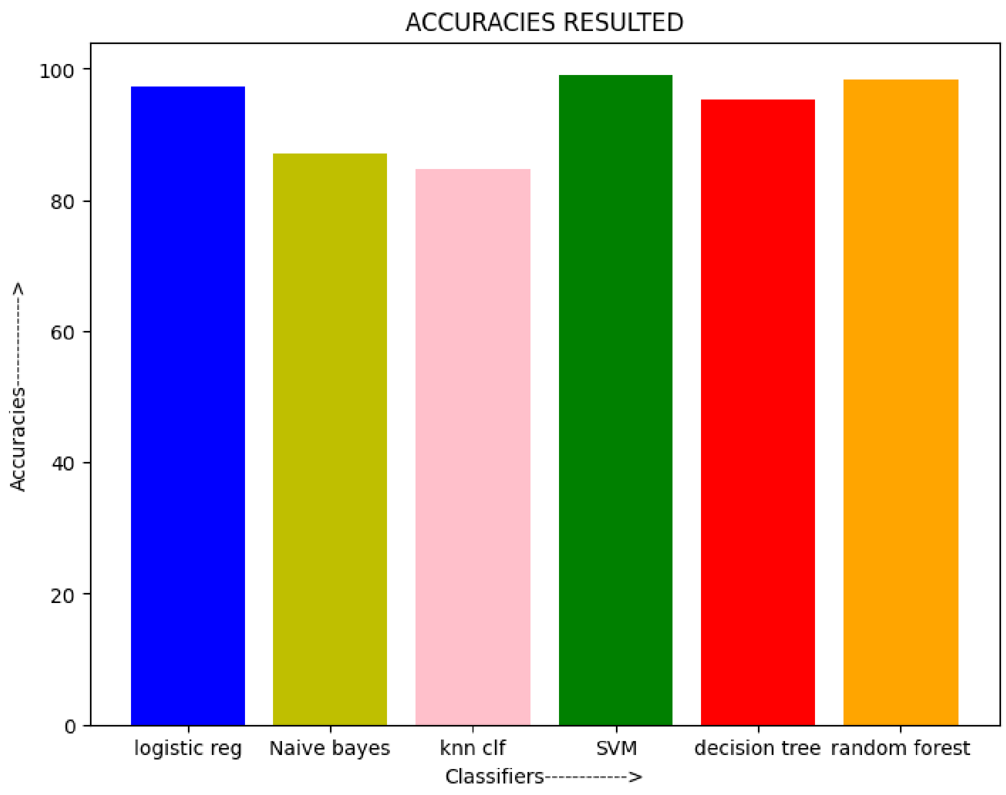
Overall, data cleaning is an important step in preparing the dataset for sculpture detection. By removing errors and inconsistencies, and ensuring that the data is accurate and complete, the model can be trained on high-quality data that will lead to more accurate and reliable predictions.

3.3. Data Visualization

Data visualization can be a powerful tool for detecting tomato leaf diseases. Some possible visualizations that can be used:

1. Heatmap: A heatmap can be used to show the severity of the disease on each leaf. This can be achieved by assigning different colors to different levels of severity.
2. Line graph: A line graph can be used to track the progression of the disease over time. This can help to identify patterns and trends in the spread of the disease.
3. Scatter plot: A scatter plot can be used to plot the size of the tomato plants against the severity of the disease. This can help to identify any correlations between the size of the plant and the severity of the disease.
4. Bar chart: A bar chart can be used to compare the severity of the disease across different plants. This can help to identify which plants are most affected by the disease.
5. Box plot: A box plot can be used to show the distribution of severity levels across different plants. This can help to identify any outliers or unusual patterns in the data.

Overall, the choice of visualization will depend on the specific data being analyzed and the insights that are being sought. It may be necessary to use multiple visualizations in order to fully understand the data and detect any patterns or trends that may be present.

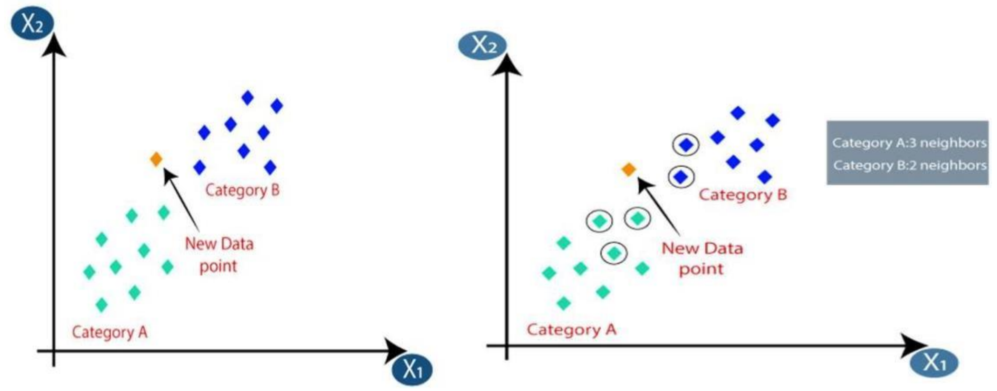


4. METHODOLOGY

4.1. Procedure to solve the given problem

LOGISTIC REGRESSION:

The K-NN working can be explained on the basis of the below algorithm:
Step-1: Select the number K of the neighbors
Step-2: Calculate the Euclidean distance of K number of neighbors
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
Step-4: Among these k neighbors, count the number of the data points in each category.
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
Step-6: Our model is ready.
Suppose we have a new data point and we need to put it in the required category. Consider the below image:



Firstly, we will choose the number of neighbors ,so we will choose the k=5.Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between points, which we have already studied in geometry. By calculating the Euclidean distance, we get the nearest.Neighbors,as three nearest neighbors in category A and two Nearest neighbors in category B.
As we can see the 3 nearest neighbors are from category A,hence this new data point must belong to categ

KNN (K-Nearest neighbours):

- K-Nearest Neighbor (KNN) Algorithm for Machine Learning
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready

CROSS VALIDATION:

Cross-validation is a technique used to evaluate the performance of a machine learning model. In k-nearest neighbors (k-NN) algorithm, cross-validation can be used to determine the optimal value of k, which is the number of nearest neighbors to consider for making a prediction.

In k-fold cross-validation, the data is divided into k equal-sized subsets. The algorithm is trained on k-1 of these subsets and tested on the remaining subset. This process is repeated k times, with each subset serving as the testing data once. The results of the k iterations are then averaged to produce a single performance metric.

To use cross-validation in k-NN, you would first split your data into training and testing sets. Then, you would use k-fold cross-validation on the training data to determine the optimal value of k. For each iteration of the cross-validation, you would train the k-NN model on the training data, with a different value of k, and then evaluate its performance on the validation set. After running all k iterations, you would select the value of k that resulted in the best performance, and use that value to train a final model on the entire training set. Finally, you would evaluate the performance of the final model on the testing set.

It is important to note that cross-validation can be computationally expensive, especially for large datasets or high values of k. However, it is a powerful technique for selecting the optimal value of k and ensuring that your model is not overfitting to the training data.

DECISION TREE:

- The model is a form of supervised learning, meaning that the model is trained and tested on a set of data that contains the desired categorization.
- A decision tree typically starts with a single node, which branches into possible outcomes.
- Each of those outcomes leads to additional nodes, which branch off into other possibilities.
- This gives it a treelike shape.
- There are three different types of nodes: chance nodes, decision nodes, and end nodes.
- The main components of a decision tree include a root node, decision nodes, chance nodes, alternative branches, and an endpoint node.
- Optional features include rejected alternatives.

RANDOM FOREST:

- Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result.
- Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.
- The random forest is a classification algorithm consisting of many decisions trees.
- It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.
- It is a set of Decision Trees.
- Each Decision Tree is a set of internal nodes and leaves.
- In the internal node, the selected feature is used to make decision how to divide the data set into two separate sets with similar responses within.

SVM (Support Vector Machine):

- Support Vector Machine Algorithm
- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

NAVIE BAYES:

Naïve Bayes is a simple probabilistic machine learning algorithm that is commonly used for classification tasks. It is based on Bayes' theorem, which describes the probability of an event given prior knowledge of related events. In Naïve Bayes, the input variables are assumed to be independent of each other, which simplifies the computation of the probability of the output variable given the input variables.

The algorithm works by first training a model on a labeled dataset, which involves calculating the probabilities of each input variable given each class label. These probabilities are then used to calculate the probability of each class label given a set of input variables using Bayes' theorem. The class label with the highest probability is then assigned to the input variables.

Naïve Bayes is particularly useful for text classification tasks, such as spam filtering, sentiment analysis, and topic classification. It can also be used for other classification tasks, such as image classification, as long as the assumption of input variable independence holds.

One of the advantages of Naïve Bayes is that it is relatively fast and requires less training data compared to other more complex machine learning algorithms. However, its performance may be limited by the assumption of input variable independence, which may not hold in some real-world applications.

Confusion Matrix:

A confusion matrix is a performance evaluation tool used in machine learning and classification tasks. It is a table that summarizes the performance of a classification model by comparing the predicted labels with the actual labels of a set of data.

The confusion matrix typically consists of four values: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

- True positives (TP) represent the number of correct positive predictions made by the model.

- True negatives (TN) represent the number of correct negative predictions made by the model.

- False positives (FP) represent the number of incorrect positive predictions made by the model.

- False negatives (FN) represent the number of incorrect negative predictions made by the model.

These values can be used to calculate various performance metrics such as accuracy, precision, recall, and F1 score.

Accuracy is the proportion of correctly classified data samples out of the total number of samples. Precision measures the proportion of true positive predictions out of all positive predictions made by the model. Recall measures the proportion of true positive predictions out of all actual positive samples. F1 score is a harmonic mean of precision and recall.

The confusion matrix is a useful tool to assess the performance of a classification model and can help identify where the model is making errors. It can be especially useful when the classes are imbalanced or when the cost of false positives and false negatives is different.

Representation:

[[TP FN]

[FP TN]]

User Modules:

matplotlib.pyplot

sklearn.model_selection

sklearn.neighbors

sklearn.naive_bayes

sklearn.metrics

sklearn sklearn.tree

sklearn.ensembleos

numpy

- matplotlib.pyplot: matplotlib.pyplot is a Python library used for creating visualizations like graphs, charts, and plots. It is a part of the matplotlib library, which is one of the most widely used data visualization librariesin Python.
- sklearn.model_selection: sklearn.model_selection is a module in the scikit-learn library that provides tools for splitting datasets into trainingand testing sets, cross-validation, and hyperparameter tuning.
- sklearn.neighbors: sklearn.neighbors is a module in the scikit-learn librarythat provides tools for implementing various types of nearest neighbor algorithms, including K-Nearest Neighbors (KNN).
- sklearn.naive_bayes: sklearn.naive_bayes is a module in the scikit-learnlibrary that provides tools for implementing Naive Bayes classifiers, a family of probabilistic classifiers based on Bayes' theorem.
- sklearn.metrics: sklearn.metrics is a module in the scikit-learn library thatprovides tools for evaluating the performance of machine learning models. It includes a wide variety of metrics for classification, regression,and clustering tasks.

- Sklearn: scikit-learn, also known as sklearn, is a popular open-source library for machine learning in Python. It provides a wide range of toolsfor various tasks in machine learning, including data preprocessing, feature selection, model selection, and evaluation.
  the examples for sklearn
  1. Loading and preprocessing datasets
  2. Splitting data into training and testing sets
  3. Creating and training machine learning models, such as decision trees, logistic regression, K-Nearest Neighbors, and Naive Bayesclassifiers
  4. Tuning hyperparameters of machine learning models using gridsearch or random search
  5. Evaluating the performance of machine learning models using various metrics such as accuracy, precision, recall, and F1-score
  6. Visualizing data and results using various plotting functions andtools
- sklearn.tree: sklearn.tree is a module in the scikit-learn library that provides tools for implementing decision tree-based models, includingdecision trees, random forests, and gradient boosting trees.
- sklearn.ensemble: sklearn.ensemble is a module in the scikit-learn librarythat provides tools for implementing ensemble methods, which are machine learning techniques that combine multiple models to improve their performance.
- os: os is a built-in Python module that provides a way to interact with theoperating system. It provides a simple and consistent interface to work with various aspects of the operating system, such as files, directories, processes, and environment variables.
- Here are some common use cases of os:
  1. File and directory operations: os provides functions for creating, deleting, renaming, and moving files and directories. For example,you can use os.mkdir() to create a new directory, os.listdir() to list the contents of a directory, and os.rename() to rename a file or directory.
  2. Path operations: os.path submodule provides functions for workingwith file and directory paths in a platform-independent way.

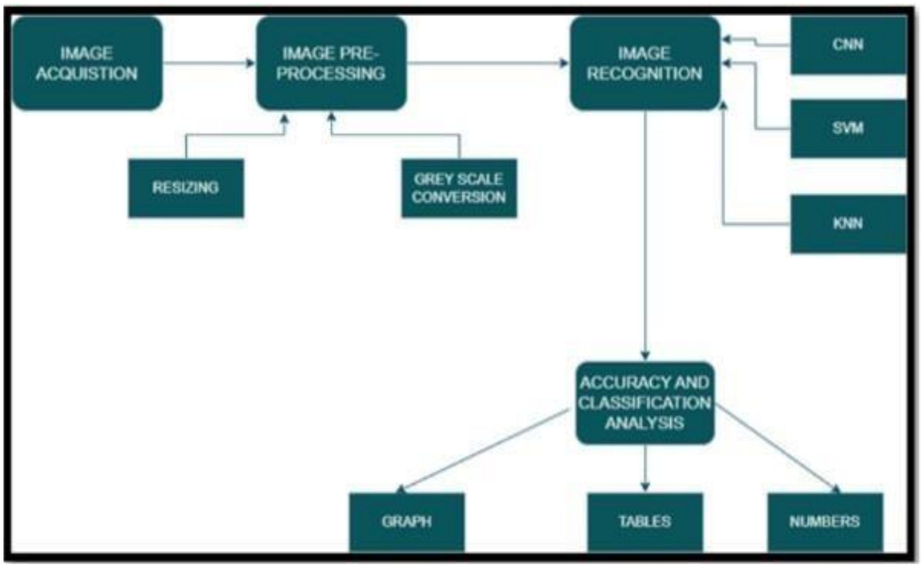
3. For example, you can use os.path.join() to concatenate two or morepath components, os.path.basename() to get the base name of a fileor directory, and os.path.abspath() to get the absolute path of a file or directory.

4. Environment variables: os.environ is a dictionary-like object that provides access to the environment variables of the current process.You can use os.environ.get() to get the value of a specific environment variable, and os.environ.setdefault() to set a default value if the variable is not defined.

5. Process operations: os provides functions for working with processes, such as os.system() to execute a command in a subshell,os.fork() to create a new process, and os.kill() to send a signal to a process.

- Numpy: NumPy (Numerical Python) is a Python library that is used for scientific computing. It provides a high-performance multidimensional array object, as well as tools for working with these arrays. Here are somecommon use cases of NumPy.

4.2. MODEL ARCHITECTURE



4.3. Software Description:

REQUIREMENTS(S/W&H/W)

HARDWARE REQUIREMENT:

System	: Ryzen5 ,5000 series
RAM	: 8gb or above
Hard disk	: 10GB OR above
Input	: keyboard and mouse
Output	: monitor or pc

SOFTWARE REQUIREMENTS:

OS	: windows 11
Platform	: jupyter notebook,goolgle colab
Program language	: python

5.Results and Discussions

Logistic Regression:

```
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(ytest,y_pred)

print("Confusion matrix:\n",cm)
```

```
Confusion matrix:
 [[112  10]
 [ 18 110]]
```

KNN Regression:

```
from sklearn.metrics import confusion_matrix
knns=confusion_matrix(ytest,y_pred)
print("Confusion matrix:\n",knns)
```

```
Confusion matrix:
 [[109 13]
 [ 29  99]]
```

NAIVE BAYES CLASSIFIER:

```
from sklearn.metrics import confusion_matrix
nbb=confusion_matrix(ytest,y_pred)
print("Confusion matrix:\n",nbb)
```

```
Confusion matrix:
 [[99  23]
 [ 13 115]]
```

SUPPORT VECTOR MACHINE:

```
from sklearn.metrics import confusion_matrix
SVMS=confusion_matrix(ytest,y_pred)
print("Confusion matrix:\n",SVMS)
```

```
Confusion matrix:
 [[118  4]
 [ 1 127]]
```

DECISION TREE CLASSIFIER:

```
from sklearn.metrics import confusion_matrix
dtcs=confusion_matrix(ytest,y_pred)
print("Confusion matrix:\n",dtcs)
```

```
Confusion matrix:
 [[102  20]
 [ 26 102]]
```

RANDOM FOREST:

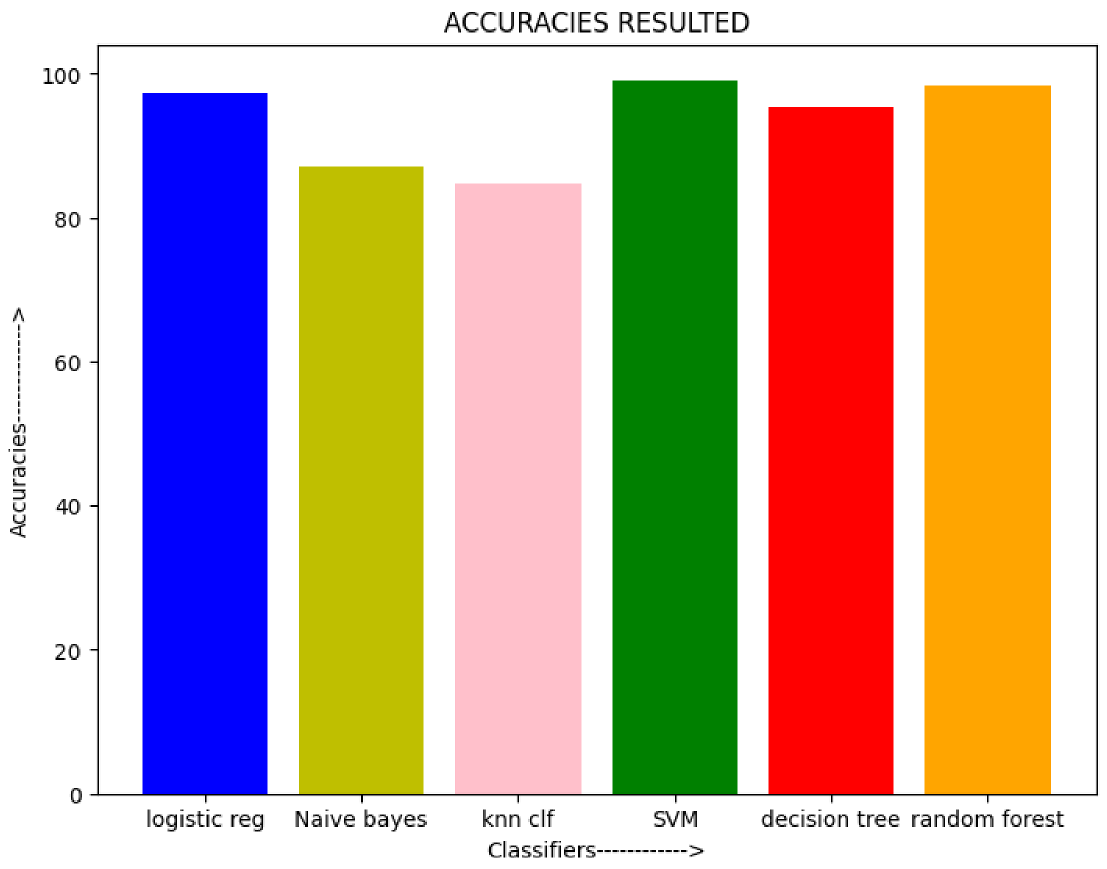
```
from sklearn.metrics import confusion_matrix
rfcs=confusion_matrix(ytest,y_pred)
print("Confusion matrix:\n",rfcs)
```

```
Confusion matrix:
 [[112  10]
 [ 6 112]]
```

Accuracy scores of Machine learning algorithms:

```
from sklearn.metrics import accuracy_score
accuracy_model = accuracy_score(y,model.predict(sc_x.transform(x)))
print("Logistic regression:",accuracy_model)
accuracy_nb = accuracy_score(y,nb.predict(sc_x.transform(x)))
print("navie bayes;",accuracy_nb)
accuracy_knn = accuracy_score(y,knn.predict(sc_x.transform(x)))
print("KNN:",accuracy_knn)
accuracy_SVM = accuracy_score(y,SVM.predict(sc_x.transform(x)))
print("Support vector machine:",accuracy_SVM)
accuracy_dtc = accuracy_score(y,dtc.predict(sc_x.transform(x)))
print("Descision tree:",accuracy_dtc)
accuracy_rfc = accuracy_score(y,rfc.predict(sc_x.transform(x)))
print("Random forest:",accuracy_rfc)
```

```
Logistic regression: 0.972
navie bayes; 0.87
KNN: 0.848
Support vector machine: 0.991
Descision tree: 0.954
Random forest: 0.984
```



This graph represents the accuracies.

RESULTS:

MACHINE LEARNING ALGORITHMS	**ACCURACY**
LOGISTIC REGRESSION	Accuracy: 0.888
K-NEAREST NEIGHBOR	Predicted value for training value 0.8533333333333334 Predicted value for testing value 0.832 Overall Accuracy: 0.848
SUPPORT VECTOR MACHINE	Training Accuracy 0.9946666666666667 Testing Accuracy 0.98 Overall Accuracy: 0.991
DECISION TREE	Training Accuracy 1.0 Testing Accuracy 0.816 Overall Accuracy: 0.954
RANDOM FOREST	Training Accuracy 1.0 Testing Accuracy 0.936 Overall Accuracy: 0.984
NAÏVE BAYES	Training Accuracy 0.8746666666666667 Testing Accuracy 0.856 Overall Accuracy: 0.87

Conclusion:

Even though there are various methods for detecting and classifying leaf diseases using automatic or computer vision, research into this field has been lacking. In addition, there are few commercial options, with the exception of those focusing on the identification of leaf species via photographs.
Over the last few years, there has been tremendous progress in the performance of convolutional neural networks. The new generation of convolutional neural networks (CNNs) has shown promising results in the field of image recognition. A novel approach to automatically classifying and detecting leaf diseases from leaf images was examined through this project utilizing deep learning techniques. With an accuracy of 90%, the developed model could distinguish healthy leaves from eight diseases that could be observed visually. On the basis of this high level of performance, it becomes apparent that convolutional neural networks are highly suitable for automatic diagnosis and detection of plants.

Future scope:

The main goal for the future project is to develop a complete system comprising a trained model on the server, as well as an application for mobile phones that display recognized diseases in fruits, vegetables, leaves and other plants based on photographs taken from the phone camera. This application will aid farmers by facilitating the recognition and treatment of leaf diseases in a timely manner and help them make informed decisions when utilizing chemical pesticides.
Also, future work will involve spreading the use of the model across a wider land area by training it to detect leaf diseases on aerial photos from orchards and vineyards captured with drones, in addition to convolution neural networks for object detection.
Drones and other autonomous vehicles, such as smartphones, to be used for real-time monitoring and dynamic disease detection in large-scale open-field cultivations. A future possibility for agronomists working at remote locations could be the development of an automated pesticide prescription system that would require the approval of an automated disease diagnosis system to allow the farmers to purchase appropriate pesticides. Thus, the uncontrolled acquisition of pesticides could be severely restricted, resulting in their excessive use and misuse, with their potentially catastrophic effects on the environment.

References:

[1] https://www.intechopen.com/chapters/76494
[2] https://link.springer.com/article/10.1007/s41348-021-00500-8
[3] https://www.sciencedirect.com/science/article/pii/S1877050920306906